

DEVELOPMENT AND IMPLEMENTATION OF
FEATURE RECOGNITION ALGORITHMS FOR
PRISMATIC COMPONENTS

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Master of Technology

by

B.Venkateshwara Rao

to the
Industrial and Management Engineering Programme
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
March, 1991

TH
658.5
R18d

IMEP-1991-M-RAO-DEV

LIBRARY

acc. No. A. 112202

C E R T I F I C A T E

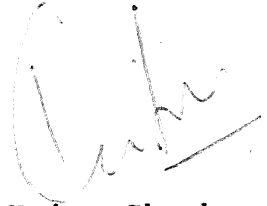
It is certified that the work contained in the thesis entitled "*Development and Implementation of Feature Recognition Algorithms for Prismatic Components*", by "*B.Venkateshwara Rao*", has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.



(Dr. S.G.Dhande)

Professor

Mechanical Engineering and
Computer Science & Engineering
IIT Kanpur



(Dr. Kripa Shanker)

Professor

Industrial and Management
Engineering Programme
IIT Kanpur

March, 1991

ACKNOWLEDGEMENT

I take this opportunity to express my gratitude towards my thesis supervisors for their constant guidance and valuable suggestions throughout my thesis period. I am thankful to Dr. Kripa Shanker for introducing me to the field of CAPP and to Dr. Dhande for teaching me the concepts of CAD. Their joint guidance was a true CAD/CAM integration for me.

Besides enabling me to assimilate part of their vast knowledge, working with my supervisors helped me to develop certain personality traits like professionalism and self confidence.

I am thankful to my classmates and other friends, whose constant encouragement and useful discussions helped me throughout my stay at IIT Kanpur. I cannot forget the company of Prusty and Sukhram for their constant entertainment, which kept me away from becoming sick of program writing.

Finally, I would like to mention my sincere thanks to all my teachers for teaching me the courses at which they excel.

B. Venkateshwara Rao
(B. Venkateshwara Rao)

March, 1991

ABSTRACT

A methodology of computer aided process planning using a feature extraction scheme has been developed in the present work. Algorithms have been developed to define the cavity volume and ghost volumes of the component, using boundary representation of the geometry of 2.5-D and 3-D prismatic parts. This approach allows the designer to explicitly define the boundary of the volume to be removed by successive machining processes. The feature recognition algorithms analyze the geometry of the cavity/ghost volume and enumerates the features that are machinable. The methodology has been implemented for 2.5-D as well as 3-D prismatic components. In case of 2.5-D geometries, the methodology includes the selection of processes and tools. In case of 3-D components, the method has been developed to identify features that are machinable. Furthermore, in case of 3-D components the solid volume is scanned in a manner that allows identifying a set of ghost volumes as a series of 2.5-D slices or slabs which are machinable by an end mill cutter. Soft ware has been developed, based on the proposed methodology. Illustrative examples of 2.5-D as well as 3-D geometries are presented.

CONTENTS

	Page
LIST OF FIGURES	viii
LIST OF TABLES	xvi
CHAPTER I: INTRODUCTION	1
1.1 PROCESS PLANNING	1
1.1.1 FUNCTIONS IN PROCESS PLANNING	1
1.1.2 COMPUTER AIDED PROCESS PLANNING	2
1.1.3 APPROACHES TO CAPP	3
1.1.3.1 VARIANT PROCESS PLANNING	3
1.1.3.2 GENERATIVE PROCESS PLANNING	5
1.1.4 RESEARCH TRENDS IN PROCESS PLANNING	7
1.2 REVIEW OF FEATURE RECOGNITION SYSTEMS	12
1.2.1 VOLUME DECOMPOSITION METHODS	12
1.2.2 SEARCHING FOR CSG PATTERNS	14
1.2.3 SEARCHING FOR FACE PATTERNS	14
1.3 SCOPE OF THE PRESENT WORK AND ORGANIZATION OF THE THESIS	15
CHAPTER II : FEATURE RECOGNITION OF 2.5-D PRISMATIC PARTS	17
2.1 METHODOLOGY	17
2.2 ALGORITHM FOR FEATURE RECOGNITION	19
2.3 DEFINITION OF FEATURE PATTERNS	23
2.3.1 T-SLOT	23
2.3.2 DOVETAIL	24
2.3.3 V-SLOT	24
2.4 ALGORITHM FOR THE EXTRACTION OF PARTIAL FEATURES	25
2.5 CHECKING FOR THE MACHINABILITY OF FEATURES	27
2.5.1 T-SLOT AND T-SLOT MILLING CUTTER	28
2.5.2 DOVETAIL AND DOVETAIL MILLING CUTTER	29

2.5.3	REVERSE DOVETAIL AND REVERSE DOVETAIL MILLING CUTTER	29
2.5.4	V-SLOT AND V-SLOT MILLING CUTTER	30
2.5.5	PARTIAL DOVETAIL AND DOVETAIL CUTTER	30
2.5.6	PARTIAL REVERSE DOVETAIL AND REVERSE DOVETAIL CUTTER	31
2.5.7	PARTIAL REVERSE DOVETAIL AND SINGLE ANGLE CUTTER	31
2.5.8	PARTIAL T-SLOT AND T-SLOT MILLING CUTTER	32
2.6.	MERGING PARTIAL FEATURES	33
2.6.1	MERGING OF TWO PARTIAL DOVETAILS	33
2.6.2	MERGING OF TWO PARTIAL REVERSE DOVETAILS	33
2.6.3	MERGING OF PARTIAL T-SLOTS	33
CHAPTER III :	FEATURE RECOGNITION OF 3-D PRISMATIC PARTS	60
3.1	METHODOLOGY	62
3.2	ALGORITHM FOR THE EXTRACTION OF GHOST VOLUMES	62
3.2.1	TRANSFORMATION OF THE PRIMITIVES TO WCS	65
3.2.1.1	TRANSFORMATION OF CUBOID2, WEDGE AND CYLINDER	66
3.2.1.2	TRANSFORMATION OF PARTIAL CYLINDER PRIMITIVES	68
3.2.2	ADJOINING ANALYSIS	69
3.2.3	MERGING OF A GROUP OF CUBOIDS WITH SAME Z-EXTENTS	70
3.2.4	MERGING OF TWO SOLIDS ALONG Z-AXIS	71
3.2.5	FINDING THROUGH AND BLIND HOLES	72
3.2.6	FINDING THE SET DIFFERENCE OF TWO SOLIDS	73
3.2.7	TOOL ACCESSIBILITY ANALYSIS	74

CHAPTER IV : SOFTWARE DEVELOPMENT AND EXAMPLES	100
4.1 IMPLEMENTATION FOR 2.5-D PARTS	100
4.1.1 INPUT MODULE	100
4.1.2 FEATURE RECOGNITION AND MACHINABILITY ANALYSIS MODULE	100
4.1.2.1 DATABASE FOR T-SLOT MILLING CUTTERS	103
4.1.2.2 DATABASE FOR DOVETAIL MILLING CUTTERS	103
4.1.2.3 DATABASE FOR REVERSE DOVETAIL MILLING CUTTERS	103
4.1.2.4 DATABASE FOR SINGLE ANGLE MILLING CUTTERS	103
4.1.2.5 DATABASE FOR EQUAL ANGLE MILLING CUTTERS	104
4.1.3 OUTPUT MODULE	104
4.2 IMPLEMENTATION FOR 3-D PARTS	104
4.2.1 INPUT MODULE	104
4.2.2 FEATURE RECOGNITION MODULE	105
4.2.3 OUTPUT MODULE	106
4.3 AN EXAMPLE OF A 2.5-D PART	106
4.4 USER MANUAL FOR 2.5-D FEATURE RECOGNITION SYSTEM	106
4.4.1 PART MODELING	107
4.5 AN EXAMPLE OF A 3.5-D PART	110
4.6 USER'S MANUAL FOR 3-D FEATURE RECOGNITION SYSTEM	110
4.6.1 PART MODELING	111
4.7 ADDITION EXAMPLE	112
CHAPTER V : CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK	126
REFERENCES	129

LIST OF FIGURES

Figure No.	Title	Page
2.1	A 2.5-D part and its AutoCAD model	
	2.1a. 2.5-D part	34
	2.1b. AutoCAD model	34
2.2	Cavity volume and ghost volumes of the part in Fig 2.1a.	35
2.3	Schematic representation of Steps 1,2,3 and 4 of feature recognition algorithm	36
2.4	Polygon of ghost volume corresponding to base plane in Fig 2.3	37
2.5	Schematic representation of Step 5 of feature recognition algorithm	
	2.5a. Ghost volume polygon before the extraction of features	38
	2.5b. Ghost volume polygon after the extraction of complete features	38
	2.5c. Ghost volume polygon after the extraction of complete and partial features	38
	2.5d. Division of ghost volume polygon into horizontal slots	39
	2.5e. Division of ghost volume polygon into vertical slots	40
2.6	Mergable (matching) partial dovetails	41

2.8	Dovetail as a pattern of 4-point sequence	41
2.9	V-slot as a pattern of 3-point sequence	42
2.10	A ghost volume polygon with stationary points	42
2.11	Schematic illustration of the extraction of partial dovetails	
	2.11a. A partial left dovetail	43
	2.11b. A partial left reverse dovetail	43
	2.11c. A partial right dovetail	43
	2.11d. A partial right reverse dovetail	43
2.12	A ghost volume polygon with stationary points after the extraction of complete features and partial dovetails	44
2.13	Schematic illustration of the extraction of partial T-slots	
	2.13a. Partial left T-slots	45
	2.13b. A partial right T-slot	45
2.14a	A T-slot and its geometry	46
2.14b	A T-slot milling cutter and its geometry	46
2.15	Machinable T-slots	
	2.15a. Case 1	47
	2.15b. Case 2	47
	2.15c. Case 3	47
	2.15d. Case 4	47

		x
2.16a	A dovetail and its geometry	48
2.16b	A dovetail milling cutter and its geometry	48
2.17	Machinable dovetails	49
	2.17a. Case 1	
	2.17b. Case 2	49
2.18a	A reverse dovetail and its geometry	50
2.18b	A reverse dovetail cutter and its geometry	50
2.19a	A V-slot and its geometry	51
2.19b	An equal angle milling cutter and its geometry	51
2.20	Machinable V-slots	
	2.20a. Case 1	52
	2.20b. Case 2	52
2.21a	A partial left dovetail and its geometry	53
2.21b	A partial right dovetail and its geometry	53
2.22a	A partial left reverse dovetail and its geometry	54
2.22b	A partial right reverse dovetail and its geometry	54

		xi
2.23a	A partial reverse left dovetail and its geometry	55
2.23b	A single angle milling cutter and its geometry	55
2.24	Machinable partial left reverse dovetails (using single milling cutter)	
	2.24a. Case 1	56
	2.24b. Case 2	56
2.25a	Partial left T-slot and its geometry	57
2.25b	Partial right T-slot and its geometry	58
2.26	Matching partial dovetails and their geometry	58
2.27	Matching partial reverse dovetails and their geometry	58
2.28	Matching partial T-slots and their geometry	59
3.1	Primitives	
	3.1a. Cuboid	76
	3.1b. Cuboid1	76
	3.1c. Wedge	76
	3.1d. Cylinder	76
	3.1e. Partial cylinder	76
3.2a	A 3-D part	77

3.2b	The 3-D part after enclosing the non-cuboids by cuboids	77
3.3	Discrete groups of cuboids of 3-D part shown in Fig 3.2a	78
3.3a.	Cuboids with Z-extents Z_1 and Z_4	
3.3b.	Cuboids with Z-extents Z_1 and Z_4	
3.3c.	Cuboids with Z-extents Z_4 and Z_5	
3.4	A 3-D part with a hole feature running in two cuboids of different Z-extents	
3.4a.	Orthogonal view	79
3.4b.	Elevation	79
3.5	The set difference of the enclosing cuboid and a partial cylinder	80
3.6	A resultant 3-D part, stock and 2.5-D part series of the resultant 3-D part	
3.6a.	Resultant 3-D part of Fig 3.2a	81
3.6b.	Stock cuboid	81
3.6c.	2.5-D part series	82
3.7	A cylinder with +ve X-axis extrusion	83
3.8	A cylinder with -ve X-axis extrusion	83
3.9	A cylinder with +ve Y-axis extrusion	83
3.10	A cylinder with -ve Y-axis extrusion	83
3.11	A partial cylinder with +ve X-axis extrusion	84

3.12	A partial cylinder with -ve X-axis extrusion	84
3.13	A partial cylinder with +ve Y-axis extrusion	85
3.14	A partial cylinder with -ve Y-axis extrusion	85
3.15	Schematic illustration of Adjoining analysis	
	3.15a. Orthogonal view	86
	3.16b. Elevation	86
3.16	A group of mergable cuboids with same z-extents	87
3.17	Merging of two cuboids of same z-extents	
	3.17a. Base planes of the solids	88
	3.17b. The counter clockwise lists of vertices of the base planes	88
	3.17c. The vertex lists after adding the intersection points of the base planes	89
3.18	Different cases of merging of cuboids along z-axis	
	3.18a. Case 1	90
	3.18b. Case 2	90
	3.18c. Case 3	91
	3.18d. Case 4	91
	3.18e. Case 5	92

3.19	Schematic illustration of Weiler-Artherton algorithm to find the set difference of 2.5-D solids	
3.19a.	2.5-D blank	93
3.19b.	2.5-D part	93
3.19c.	Base plane of 2.5-D blank	94
3.19d.	Base plane of 2.5-D part	94
3.19e.	Vertex lists of base planes	95
3.19f.	Vertex list after adding the intersection points	96
3.20	Tool accessibility of ghost volumes	
3.20a.	2.5-D stock	97
3.20b.	Ghost volume	97
3.20c.	The base planes of the stock and ghost volume	98
3.21	Tool accessibility of the holes	
3.21a.	A through hole	99
3.21b.	Blind hole opening on XY-plane, $Z=Z_1$	99
3.21c.	Blind hole opening on XY-plane, $Z=Z_2$	99
4.1	System flow chart for feature recognition of 2.5-D part	113
4.2	The program flow chart for feature recognition and machinability analysis module (2.5-D part)	114
4.3	System flow chart for feature	115

		xv
4.4	Program flow chart for feature recognition module (3-D part)	116
4.5	AutoCAD model of a 2.5-D part	117
4.6	Base plane Corresponding to Fig 4.5 with T-slots on left side	117
4.7	Base plane corresponding to Fig 4.5 with complete features	118
4.8	Base plane corresponding to Fig 4.5 with complete and partial features	118
4.9	Base plane corresponding to Fig 4.5 with all features including the slots	119
4.10	AutoCAD model of a 3-D part shown in three views	119
4.11	Alpha-numerical output of the hole information in the 3-D part shown in Fig 4.10	120
4.12	Graphical output of global the ghost volumes in Fig 4.10	120
4.13	AutoCAD model of a complicated 2.5-D part	121
4.14	Features recognized in the part shown in Fig 4.13	121
4.15	AutoCAD model of a complicated 3-D part	122

LIST OF TABLES

Table No	Title	Page
4.1	Specifications for T-slot milling cutter	123
4.2	Specifications for dovetail milling cutter	123
4.3	Specifications for reverse dovetail cutter	124
4.4	Specifications for single angle mill cutter	124
4.5	Specification for double angle milling cutter	125

Chapter I

INTRODUCTION

1.1 PROCESS PLANNING

Process planning is the systematic determination of the methods and means by which a product is to be manufactured economically. It is the subsystem responsible for the conversion of design data to work instructions. More specifically, *process planning* is a function within a manufacturing facility that establishes which machining processes and parameters are to be used to convert a piece part from its initial form predetermined from an engineering drawing.

Process planning is essential for production and also determines the efficiency of production. A detailed plan containing route, machines, processes and tools etc. required for production of the part is the output of *process planning*. This plan is called *route sheet*.

1.1.1 Functions In Process Planning

The *process planning* functions in an industry involve several or all of the following functions:

- Recognition of machining features,
- Selection of processes,
- Sequencing of operations,
- Selection of cutting tools,

- Selection of jigs and fixtures,
- Selection of process parameters, and
- Planning of tool path and generation of NC part programs.

1.1.2 Computer Aided Process Planning

Process planning requires a significant amount of experience and time. According to a study [Houtzeel,1981] a typical process planner is a person over 40 years of age, with a long term experience in a machine shop. Modern industry faces the scarcity of skilled labour to do the job. *Process planning* of complicated parts takes hours and even days. Thus it is tedious and error prone. Inconsistency is another problem with manual process planning, as different process planners perceive different processes as economical. *Process planning* has become a gargantuan task due to a large variety of products and decreasing product life cycles in modern industry.

Advent of computers in manufacturing made a considerable dent on process planning and this resulted in *Computer Aided Process Planning (CAPP)* systems. In *CAPP*, a computer does either several or all of the process planning functions. The development of NC machine tools in 50's replaced the task of generating manually cumbersome cutter path instructions by computer generated NC code. Database management systems for the storage and retrieval of process plans were developed in 60's. Computer aided systems for the selection of processes, machines tools, tools, jigs and fixtures have been developed since late 70's.

First generation CAPP systems required human intervention to process the design diagram of the object and to convert it into a computer readable form, code or specialized language. In early 80's it is realized that CAD systems provide a computer readable part description [Requicha,1980]. Human intervention can be minimized to a great extent if this part description is used as input to CAPP systems. After reading the part description, obtained from CAD drawings, the manufacturing features present in the part are recognized. This is called *Feature Recognition*, which is not a trivial problem. Most of the ongoing research in CAPP is directed towards a generic *feature recognition* system, which is very essential to the complete integration of CAD and CAM.

1.1.3 Approaches To CAPP

There are two basic approaches to CAPP - *variant* and *generative*. In *variant* approach an existing process plan of a similar component is retrieved and is edited to create its '*variant*' to suit the specific requirement of a component being planned. In *generative* approach, a new process plan is generated for each component without referring to existing plans.

1.1.3.1 Variant Process Planning

It is based on the concept that similar parts have similar process plans. There are two stages in developing a *variant process planning* system - preparatory and production.

During the preparatory stage, a coding scheme ,encompassing various characteristics and attributes like geometric shapes and process similarities, is established. The existing components are coded, classified and grouped into part families, depending on the similarities in codes. Families can be described by a set of family matrices. Each family is a binary matrix with a coloumn for each digit in code and a row for each value a code digit can take. A non-zero element in the matrix indicates that the particular digit can have the value of that row, eg. element(3,2) equals one implies that a code x3xxx can be a member of the family. A standard plan is a process plan to manufacture the entire family. This preparatory stage is a time consuming process, and can take 18-24 man years for a machine shop.

During the production stage, the system is ready to create *process plans* for new parts. A new part is coded, the code is used to retrieve the standard plan of the family to which it belongs. Then the standard plan is edited.

Variant process planning is very easy to understand, learn and use. *Process planner* has full control over the final plan. It is especially useful for manufacturing industries which produce similar components repetitively. The system cannot be used for radically different parts. Since human intervention is still required, complete automation is not possible with *variant process planning*.

1.1.3.2 Generative Process Planning

In *generative process planning* process information is synthesized to create a process plan of a new part automatically. A *generative process planning* has three components- Part description, Decision logic and algorithms, and manufacturing databases.

Part description is the front end of a *generative CAPP* system, through which, the geometry, dimensioning, and surface quality requirements for a machined part are defined. The first generation of *generative CAPP* systems used coding based on design and/or manufacturing attributes. OPITZ [Opitz,1970], DCLASS [Allen,1979], and MICLASS [OIR,1983] are such coding systems. APPAS [Wysk,1977] and GENPLAN [Tulkoff,1981] use coding systems in description modules. But coding requires considerable human intervention and exact information is lost when a part is coded by finite digits of code. This limitation led to the development of special description languages.

AUTAP system [Evershiem and Esch 1983] uses a descriptive language in which a part is described using geometric *feature* elements such as cylinders, chamfer, radius and technological elements like tolerance. In CIMS/DEC part description system, a part shape is described using volumetric elements obtained by revolving or translating of generating surfaces. Each generating surface is a concatenation of profile elements given by directed line segments. Technological information can be supplied to each profile element. Jakubowski [1982] gave a syntactic method for

part description. He used concepts of basic primitives which are segments of straight lines and curves. GARI [Descolte and Latombe, 1981] uses a part description based on a set of features, such as tapped holes, countersunk holes, bores, notches, faces etc. to describe the part. The features are described by diameter, surface finish, perpendicularity and tolerance etc..

Although codes and special languages serve the purpose of part description adequately, the considerable human intervention was an obstacle in the path of the goal towards complete automation. This led to the development of many *feature recognition* systems since early 80's. Both 2-D and 3-D CAD models have been used for *process planning* applications.

Decision logic and algorithms module imitates the decision making function of a *process planner*. These functions include all or several of *process planning* functions. The decision logic is represented using decision trees, decision tables and Artificial Intelligence techniques. Algorithms are used to perform computations and guide the system during decision making process.

Manufacturing databases module stores the manufacturing knowledge. The knowledge representation methods are related directly to the decision logic because static data is the representation and dynamic use of the data becomes the decision logic. Several databases are required to provide a *process planning* system with the information required for making decisions.

1.1.4 Research Trends In Process Planning

Earlier research in *process planning* was concentrated on the *variant* approach. In 70's group technology (GT) based *variant process planning* systems were developed. Those systems could neither make decisions nor help in making decisions, since there was no intelligence involved in them. But trends in manufacturing since 80's have shown a demand for a large variety of parts. As the *variant process planning* can only work in a machine shop involving the production of similar parts, it could not meet the requirements and focus of research shifted towards the *generative process planning*.

Research in the first generation of *generative process planning*, i.e. in early 80's, was directed at the development of better methods for representation of decision logic and manufacturing databases. A *generative process planning* system contains tremendous amount of knowledge - rules in decision making and facts about the machine shop. These rules need updating with the rapid advances in manufacturing technologies. Traditional systems do not allow the updating of facts and rules as they are coded line by line in the program statements. This rigidity is overcome by the application of Artificial Intelligence (AI) and expert systems. An expert system stores the knowledge in a special manner so that it is possible to add, delete, and modify rules and facts in the knowledge base without rewriting the program.

The declarative facts in an expert system are the knowledge about machine tools, cutting tools, jigs and fixtures, and machining operations. Procedural rules store the decision logic involved in the selection of operations, machine tools, cutting tools, jigs and fixtures, operation sequences and machining parameters.

Although there are many ways of representing declarative facts, frames are found to be the best way of representation [Wang and Wysk,1989]. A frame consists of name and several attributes, where each attribute has an associated value. The name identifies an object and attributes define the characteristics of the object. An attribute may contain several features and each feature can contain several sub features, and so on.

For instance, knowledge about a screw-cutting lathe is represented in the form of a frame as follows :

Machine name : M

List of attributes :

1.length_of_bed = 120

2.swing_of_lathe = 30

3.no_of_turrets = 1

4.turret orientation = side

5.rotating_turret_spindle = 1

6.chucking_capability :

1.manual_chucking = 1

2.bar_feeder = 0

```

3.tail_stock = 1
7.tools_available :
    1.r_hand = 1
    2.l_hand = 1

```

Production rule is the best way of representing a procedural rule [Wang and Wysk,1989]. A production rule is expressed in the form of a condition/action pair. The condition part of the rule is checked every time a rule is selected for execution.

For instance, a rule about the selection of slab milling may be represented as follows :

```

IF { feature is slot and
      slab mill cutter is available and
      surface finish_required lies in the range of
      surface finish of slab milling}
THEN { select the slab milling to machine the slot}.

```

Machine learning, generation of multiple process plans, reasoning explanation and process planning under uncertainty are the future research topics in AI-based CAPP systems. A system should be able to accumulate the knowledge if user is more knowledgeable than the system. However, the incorrect and inconsistent information should be checked. Alternative processes can be employed to machine a feature and different sequences of processes can be generated to machine a part. This results in multiple process plans. An expert system with reasoning capability can explain how it arrived at a particular decision.

Uncertainty should be included in decision making if process planning is to be integrated with scheduling.

Earlier *generative process planning* systems used either codes or special languages to describe a part. Part description in these methods required human efforts to process the design drawing. Since mid-80's, CAD models are being used to supply part description to CAPP systems. Part description in CAD models is in the form of basic geometry, such as vertices, edges, faces, primitive solids etc., and topology which is unsuitable for direct application to *process planning*. Process planning needs information in the form of *features*, which need to be extracted from CAD models. This led to the development of geometric reasoning, which helps the system to understand the part geometry, shape and relationships among various features. A *feature* is a region of a part having some manufacturing significance in the context of machining. The importance of *feature recognition* stems from the fact that each *feature* can be planned for the decisions about how it is to be manufactured, and this information collectively gives the *process plan*. Feature extraction can be viewed as a fundamental aspect of machine understanding and can be extended to purposes other than manufacturing.

Another approach to *CAD/CAM integration*, and hence to CAPP, has been *feature based design* in which the designer is provided with a library of features along with a set of operators like add, delete and modify etc. to create a part model. Proponents of

feature based design argue that when a designer starts the design process, most of the features are already known to the designer and in the process of converting them to CAD model the information is lost and has to be recreated by *feature recognizers*.

The push towards *feature based design* also stems from the fact that traditional CAD systems are not capable of providing the new information required for other analysis such as, Design for Manufacturability and Functional information analysis about the part etc. But *feature based design* faces a grave problem. The set of *features* used to design a part based on a given set of machining processes, may be useless if the a manufacturing process is changed, say from turning to forging or rolling. Feature recognizers will be needed to assist the conversion of features from one application to another.

Integrating and interfacing of *process planning* with other manufacturing functions, like scheduling and shop floor control, is another major research issue. Since *route sheets* generated by *process planning* forms input to scheduling, the integration of *process planning* and scheduling is very essential [Khoshnevis and Chen, 1990]. Scheduling generates an optimal schedule based on the optimal *route sheets* generated by *process planning*. Since this hierarchical optimization may not give globally optimized schedules, the integration seems to be a desirable one.

1.2 REVIEW OF FEATURE RECOGNITION SYSTEMS

A *feature recognition* system translates the part definition data created by a CAD system into machining features needed to drive a *process planning* system. So the amount and type of available part description becomes a major consideration. The two-dimensional boundary data is sufficient for the description of axisymmetric parts and prismatic parts with linear sweep. 3-D solids are represented either as the boundary representation (B-rep) or as the constructive solid geometry (CSG) binary tree.

B-rep uses low level entities such as faces, edges, vertices and topological relationships. Objects are represented by their bounding faces. Faces are further broken down and represented by edges and vertices.

In CSG a part is represented as a tree of primitive volumes, such as block, wedge, cone, cylinder, sphere and torus , and boolean set operators. In CSG a part need not have a unique representation.

There are mainly three approaches to *feature recognition* - volume decomposition methods, searching for CSG patterns and searching for face patterns

1.2.1 Volume Decomposition Methods

These methods seek to segment the volume to be removed into subvolumes that corresponds to machining operations. The removal volume, some times called *cavity volume*, is defined as the set difference of the initial stock and the part. Each subvolume can

be associated with a machining *feature*, and therefore volume decomposition amounts to *feature recognition*.

Woo [1982] presented an algorithm for extracting volumetric features by using convex hulls. His algorithm uses B-rep as the part description, and yields a unique series expansion of an object in terms of convex volumes with alternating signs for volume addition and subtraction. The drawback of the method is that the extracted convex volumes are not necessarily machining features. CAM-I Inc. [1984], a not-for-profit organization, proposed a similar principle. The *cavity volume* is divided into individual components, called delta volumes. The delta volumes are further divided in to subdelta volumes, each having various machining techniques associated with it for removing material. These subdelta volumes represent units of work, which serve as input to algorithms for generation of cutter paths.

Dong and Wozny [1990] call the subdelta volume as feature volume and they describe three algorithms for creating feature volumes from B-rep. As an application of feature volumes for process planning, an algorithm was presented for global tool accessibility analysis. Perng, Chen and Li [1990] convert a CSG representation of a part to destructive solid geometry (DSG) representation. A DSG tree is a special kind of CSG tree in which all operations are set differences. Then the machining features are extracted from the DSG tree.

1.2.2 Searching For CSG Patterns

Woo [1977] proposed a method for extracting *features* from an object's CSG representation. He defines objects by a restrictive form of CSG, and considers volumetric features, also defined by CSG. An object's CSG representation is searched by the use of AI techniques to extract patterns, CSG fragments, that match with the feature definitions.

Lee and Fu [1984] use tree manipulation techniques to rearrange an object's CSG representation so as to group certain CSG patterns that correspond to solid features. Features are defined as CSG combinations of primitives whose axes of symmetry satisfy certain geometric relationships.

Non-uniqueness of CSG representation is a fatal drawback and hence searching for CSG patterns is not a promising approach.

1.2.3 Searching For Face Patterns

An alternative and more promising approach is defining *features* as a set of faces that satisfy certain geometric relationships. The search is done over the boundary representation of the part.

Henderson and Anderson [1984] developed procedures for searching the part description, recognizing the *cavity volumes*, extracting features from *cavity volumes* and arranging them in feature graph, a high level data structure appropriate for *process planning*. Kyprianou [1980] classifies the edges as

convex, concave or smooth. Face sets are classified as depressions or protrusions by the analysis of faces and edges.

Joshi and Chang [1989] developed a concept, which seems to be the extension of Kyprianou's work, called attributed adjacency graph (AAG) for the recognition of machined features from a 3-D boundary representation. Then graph based heuristics are applied on AAG to extract features.

1.3 SCOPE OF THE PRESENT WORK AND ORGANIZATION OF THE THESIS

The prismatic parts can be grouped into two categories - 2.5-D and 3-D parts. The 2.5-D parts can be generated by the extrusion of a plane. For a 2.5-D part, a 2-D drawing of the base plane and the extrusion length along the direction perpendicular to the base plane are adequate to describe. The 3-D parts are complicated and are represented by geometric models having B-rep or CSG representation.

The present work is aimed at the *feature recognition* of 2.5-D and 3-D prismatic parts and is based on the volume decomposition approach. AutoCAD is used to model the parts. The *feature recognition* algorithms are based on the part modeling in AutoCAD.

Chapter II starts with the part representation of 2.5-D parts in AutoCAD. It describes the methodology and the algorithm for the *feature recognition* of 2.5-D parts based on the part description.

Chapter III starts with the part modeling of 3-D parts in AutoCAD. It proceeds to explain the methodology and the algorithm for the *feature recognition* of 3-D parts based on the part modeling.

Chapter IV describes the software development and examples. It explains the *feature recognition* of 2.5-D parts and 3-D parts with one illustration each. The chapter also includes the User manual.

Chapter V gives the conclusions and limitations of the present work and suggestions for the future work.

FEATURE RECOGNITION OF 2.5-D PRISMATIC PARTS

A 2.5-D prismatic part can be represented as a base plane and its extrusion, along the direction perpendicular to the base plane. The base plane is drawn as a polyline and its extrusion is defined by the thickness of polyline. Polyline is a sequence of lines, which may or may not be a closed figure. In the present system closed polylines in which all vertices lie on a plane, $z=0$, are considered. In this system AutoCAD is used to model the parts. The AutoCAD possesses polyline as one of its drawing entities, which is used for defining the base plane. The extrusion is defined by the thickness, which is one of the properties of the drawing entities in AutoCAD. AutoCAD gives a data exchange file (DXF) file, which has geometric information of the drawing. DXF file contains polyline as a sequence of vertices and is the input to the *feature recognition* module, explained in Chapter IV. Fig. 2.1.a shows a 2.5-D part and Fig 2.1.b its representation in AutoCAD.

2.1 METHODOLOGY

From the DXF file, the geometry of the polyline is read and it is stored as a linked list of the vertices. Since the polyline considered in the system is a closed figure, it can be called as a polygon and its minimum enclosing rectangle can be easily found out. The set difference of the enclosing rectangle and the polygon gives the areas corresponding to the *cavity volume* to be removed from the stock. The assumption here is that stock is a cuboid and is the extrusion of the enclosing rectangle along the

The *cavity volume* is a collection of discrete volumes called *ghost volumes*. Fig 2.2 shows the stock and *cavity volume* corresponding to the part in Fig 2.1a. A *ghost volume* is also an extrusion of a polygon. Since in the present system right prismatic parts are considered, the extrusion of the *ghost volumes* all be along the direction perpendicular to the polygon or equivalently to the base plane. The polygon is stored as a linked list of points, where each point in the linked list corresponds to an edge in the *ghost volume* and each link corresponds to a plane. Each *ghost volume* is searched for various machining features. Features are defined as the patterns of points, which satisfy certain conditions, and the linked lists are searched for these patterns. The machining features, considered in the present system are the instances of

- (1) Slot,
- (2) T-slots (Complete t-slot and partial t-slot) and
- (3) Dovetails (Complete dovetail and partial dovetail).

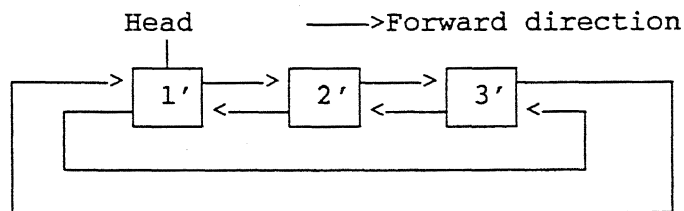
After *feature recognition*, each feature is tested for *machinability*. A feature is machinable if (i) a cutting tool is available to machine it and (ii) it can be machined without tool-part interference. A cutting tool database, containing information about T-slot cutters, dovetail cutters, reverse dovetail cutters and V-slot cutters, is used in the system. A cutting tool is said to be available, if it is contained in the tool database. If a cutting tool is available, then the geometries of the part and the cutting tool are compared to find out the tool-part interference.

2.2 ALGORITHM FOR FEATURE RECOGNITION

Feature recognition algorithm can be explained as follows:

Step 1. Read the DXF file and store the vertices of polyline in a doubly linked list. The forward direction of the linked list is defined to correspond to the clock-wise direction in the base plane.

For example, for the geometry shown in Fig 2.3 the list corresponding to vertices 1,2 and 3 would appear as shown below :



Step 2. Find the enclosing rectangle by selecting the minimum and maximum X and Y co-ordinates among all vertices, as shown in Fig 2.3.

Step 3. Find the lower left point of the base plane and call it as the head of the linked list. Lower left point, P, is defined such that P_y is minimum among all points, whose X-coordinates are equal to X_{minimum}

Step 4. Scan the linked list, starting from the head, to classify the vertices into four categories i.e. left, top, right and bottom sides. Divide the linked list into four parts one for each side. The deflection points i.e. points where side changes from left to top / from top to right / from right to bottom / from bottom to left are marked by the following rules.

- (i) IF P is the first point to satisfy $P_y = Y_{\text{maximum}}$
THEN P is the deflection point from left to top.

- (ii) IF P is the first point to satisfy $P_x = X_{\text{maximum}}$,
THEN P is the deflection point from top to right.
- (iii) IF P is the first point to satisfy $P_y = Y_{\text{minimum}}$,
THEN P is the deflection point from right to
bottom
- (iv) The lower left point is the deflection point from
bottom to left.

Step 5. Find the *ghost volumes* on all four sides from the corresponding linked lists by scanning from the start point of the list to end point of the list. The following rules are followed in the identification of ghost volumes. Assuming P be a point in the linked list, the rules for the start point of a ghost volume are :

- (i) IF { (side is left) and ($P_x = p_x^{\text{previous}} = X_{\text{minimum}}$)
and ($P_x < p_x^{\text{next}}$) and (P is a not deflection
point) }

THEN { P is start point of a *ghost volume* }.

This rule identifies the vertices 2 and 6 in Fig 2.3 as the start points of *ghost volumes*.

- (ii) IF { (side is top) and ($P_y = p_y^{\text{previous}} = Y_{\text{maximum}}$)
and ($P_y > p_y^{\text{next}}$) and (P is not deflection
point) }

THEN { P is start point of a *ghost volume* }.

This rule identifies the vertices 9 and 13 in Fig 2.3 as the start points of *ghost volumes*.

- (iii) IF { (side is right) and ($P_x = p_x^{\text{previous}} = X_{\text{maximum}}$) and ($P_x > p_x^{\text{next}}$) and (P is not a
deflection point) }

THEN { P is start point of a *ghost volume* }.

This rule identifies the vertex 17 in Fig 2.3 as the start point of a *ghost volume*.

(iv) IF { (side is bottom) and ($P_Y = p_Y^{\text{previous}} = Y_{\text{minimum}}$) and ($P_Y < p_Y^{\text{next}}$) and (P is not a deflection point) }

THEN { P is start point of a *ghost volume* }.

This rule identifies the vertex 20 in Fig 2.3 as the start point of a *ghost volume*.

The rules for the end point of a *ghost volume* are :

(i) IF { (side is left) and ($P_X = p_X^{\text{next}} = X_{\text{minimum}}$) and ($(P_X < p_X^{\text{previous}})$ or (P is a deflection point)) }

THEN { P is end point of a *ghost volume* }.

This rule identifies the vertices 5 and 8 in Fig 2.3 as the end points of *ghost volumes*.

(ii) IF { (side is top) and ($P_Y = p_Y^{\text{next}} = Y_{\text{maximum}}$) and ($(P_Y > p_Y^{\text{previous}})$ or (P is a deflection point)) }

THEN { P is end point of a *ghost volume* }.

This rule identifies the vertices 12 and 15 in Fig 2.3 as the end points of *ghost volumes*.

(iii) IF { (side is right) and ($P_X = p_X^{\text{next}} = X_{\text{maximum}}$) and ($(P_X > p_X^{\text{previous}})$ or (P is a deflection point)) }

THEN { P is end point of a *ghost volume* }.

This rule identifies the vertex 19 in Fig 2.3 as the end point of a *ghost volume*.

(iv) IF { (side is bottom) and ($P_Y = p_Y^{\text{next}} = Y_{\text{minimum}}$) and ($(P_Y < p_Y^{\text{previous}})$ or (P is a deflection point)) }

THEN { P is end point of a *ghost volume* }.

This rule identifies the vertex 23 in Fig 2.3 as the end point of a *ghost volume*.

A start point, subsequent end point and the points in between form a *ghost volume*.

Store the *ghost volumes* as linked lists of points, along with the information about the side. Fig 2.4 shows the *ghost volumes* corresponding to the part in Fig 2.3.

Step 6. Search the *ghost volumes* for the predefined patterns of machinable features by scanning from start to end. When an instance of a *feature* is encountered, the pattern of points is separated from the *ghost volume* and added to the *feature* list. The *feature* recognition is done in two steps.

- (a) Extract the complete *features*, instances of t-slots, dovetail and v-slot in that order. Fig 2.5.a shows a *ghost volume* and Fig 2.5.b shows the *ghost volume* after extracting the complete *features*.
- (b) Extract the partial *features*, partial dovetails, partial reverse dovetails, and partial t-slots. Fig 2.5.c shows the *ghost volume* after extracting the partial *features* in Fig 2.5.a.

If partial *features* were searched before complete *features*, each complete *feature* would be recognized as two partial *features*. Searching time may be reduced, if the *features* whose patterns contain more number of points are searched prior to those whose patterns contain less number of points.

- (c) Divide the *ghost volumes* into slots. This is equivalent to the division of the convex polygons into rectangles, since after the removal of *features* by Steps (a) and (b) each *ghost volume* is an extrusion of a convex polygon. Each rectangle is a base plane of a slot. Fig 2.5.d and Fig 2.5.e show

the two ways of dividing the *ghost volume* into slots. In the implementation of the algorithm, the steps (a), (b) and (c) are coded with respect to top side. They are applied to other sides by rotating the *ghost volumes* appropriately. i.e. *ghost volumes* on the left are rotated by -90 degrees about the lower right corner of the stock, *ghost volumes* on the right are rotated by 90 degrees about the lower left corner of the stock and *ghost volumes* on bottom are rotated by 180 degrees about the upper left corner.

Step 7. Merge the matching partial features into complete features. eg. If a partial left t-slot has a matching partial right t-slot then they are unified in to a t-slot. Similar rule applies for partial dovetails and reverse partial dovetails. Fig 2.6. shows a *ghost volume* with matching partial dovetails - partial left dovetail (pldt) and partial right dovetail (prdt).

Step 8. Check the *machinability* of the features by checking two conditions - cutting tool availability in cutting tool database and tool-part interference.

2.3 DEFINITION OF FEATURE PATTERNS

The patterns for complete features are defined as the sequences of points which satisfy certain necessary and sufficient conditions. The patterns are defined with respect to top side.

2.3.1 T-Slot

As shown in Fig 2.7, a t-slot is a sequence of eight points. Assuming X is the array of X-coordinates and Y is the array of Y-coordinates of the eight point sequence, the t-slot then

satisfies the following conditions.

```
( X[1] = X[2] and Y[1] > Y[2] ) and
( X[2] > X[3] and Y[2] = Y[3] ) and
( X[3] = X[4] and Y[3] > Y[4] ) and
( X[4] < X[5] and Y[4] = Y[5] ) and
( X[5] = X[6] and Y[5] < Y[6] ) and
( X[6] > X[7] and Y[6] = Y[7] ) and
( X[7] = X[8] and Y[7] = Y[8] ) and
( (X[2] - x[3]) = (X[6] - x[7]) ) and
( (Y[3] - Y[4]) = (Y[6] - Y[5]) )
```

2.3.2 Dovetail

As shown in Fig 2.8, a dovetail is a sequence of four points. Assuming X is the array of X-coordinates and Y is the array of Y-coordinates of the four point sequence, the dovetail then satisfies the following conditions.

```
( X[1] > X[2] and Y[1] > Y[2] ) and
( X[2] < X[3] and Y[2] = Y[3] ) and
( X[3] > X[4] and Y[3] < Y[4] ) and
( Y[1] = Y[4] )
```

2.3.3 V-Slot

As shown in fig 2.9, a v-slot is a sequence of three points. Assuming X is the array of X-coordinates and Y is the array of Y-coordinates of the three point sequence, the v-slot then satisfies the following conditions.

```
( X[1] > X[2] and Y[1] > Y[2] ) and
( X[2] < X[3] and Y[2] < Y[3] ) and
( Y[1] = Y[3] and (X[2] - X[1]) = (X[3] - X[2]) )
```

The above conditions are used in the *feature extraction* of the complete features.

2.4 ALGORITHM FOR THE EXTRACTION OF PARTIAL FEATURES

The following algorithm is followed for the recognition of partial features in the *ghost volumes* from which the complete features have already been extracted.

Step 1. Scan the *ghost volume* to find out the stationary points. Stationary point, P_s , is the point, at which $\Delta Y = (P_Y^{\text{next}} - P_Y)$ changes its sign. P_s^+ is a point where the sign of ΔY changes from - to + and P_s^- is a point where the sign changes from + to -. Classify the start end points of the *ghost volume* as P_s^- s. Every P_s^- , except the last one, is followed by a P_s^+ . Fig. 2.10 shows a *ghost volume* with P_s s.

Step 2. Scan the *ghost volume* from star to end. While scanning between P_s^- and P_s^+ the following rules are applied to extract the left partial features.

(i) IF { $P_x > P_x^{\text{next}}$ and $P_y > P_y^{\text{next}}$ }

THEN { P is start point of a partial left dovetail }.

Draw a perpendicular from P on to a line, parallel to X-axis, through P^{next} , as shown in Fig 2.11.a. Let Q be the intersection point. P, Q and P^{next} form a partial left dovetail.

(ii) IF { $P_x < P_x^{\text{next}}$ and $P_y > P_y^{\text{next}}$ }

THEN { P is start point of a partial left reverse dovetail }

Draw a perpendicular from P^{next} on to a line, parallel to X-axis, through P, as shown in Fig 2.11.b. Let Q be the intersection point. Q, P^{next} and P form a partial left reverse dovetail.

While scanning from P_s^+ to P_s^- , the following rules

are applied to extract right partial features.

(i) IF { $P_x > p_x^{\text{next}}$ and $P_y < p_y^{\text{next}}$ }

THEN { P is start point of a partial right dovetail }

Draw a perpendicular from p^{next} on to a line, parallel to X-axis, through P, as shown in Fig 2.11.c. Let Q be the intersection point. p^{next} , Q and P form a partial right dovetail.

(ii) IF { $P_x < p_x^{\text{next}}$ and $P_y < p_y^{\text{next}}$ }

THEN { P is start point of a partial right reverse dovetail }

Draw a perpendicular from P on to a line, parallel to X-axis, through p^{next} , as shown in Fig 2.11.d. Let Q be the intersection point. Q, P and p^{next} form a partial right reverse dovetail.

Step 3. After the extraction of partial dovetails and partial reverse dovetails, the ghost volumes consist of lines parallel to X and Y axes. A ghost volume contains partial t-slots, if it contains concave corners, as shown in Fig 2.12. Searching for partial t-slots is similar to the search of partial dovetails. After finding the P_c s, the following rules are applied to extract partial t-slots.

While scanning from P_c^- to P_c^+ in forward direction,

IF { $P_x > p_x^{\text{next}}$ }

THEN { P is start point of a list of points which contains one or more partial left-slots }

If Q is the next point in the list such that, $\{Q_x \geq P_x\}$, a perpendicular line is drawn from P on to the line, parallel to X-axis, through Q, as shown in

Fig 2.13.a. Let R be intersection point. The linked list from P to R is divided in to as many partial left t-slots as the number of steps in it.

While scanning from P_C^+ to P_C^- in back ward direction,

IF { $P_x < P_x^{\text{next}}$ }

THEN { P is start point of a list of points
which contains one or more partial
right t-slots }

If Q is the next point in the list such that, $\{Q_x \leq P_x\}$, draw a perpendicular line is from P on to the line, parallel to X-axis, through Q, as shown in Fig 2.13.b. Let R be intersection point. The linked list from P to R is divided in to as many partial right t-slots as the number of steps in it.

2.5 CHECKING FOR THE MACHINABILITY OF FEATURES

To check the machinability of various features a database of cutting tools is used in the system. The types of cutting tools, considered in the system, are

1. T-slot mill cutter is used to machine t-slot and partial t-slot.
2. Dovetail mill cutter is used to machine dovetails and partial dovetails.
3. Reverse dovetail mill cutter is used to machine reverse dovetails and partial reverse dovetails.
4. Equal angle mill cutter is used to machine v-slots.
5. Single angle mill cutter to machine partial reverse dovetails.

Tools of type 1,2,and 3 use vertical spindle, where as tools of type 4 and 5 use horizontal spindle.

The tool database stores the geometry of the cutting tools. After extracting the *features* from the part, the *feature* geometry is used to search the cutting tool database for a suitable tool. If a cutting tool is available, then its geometry is compared with the geometry of the corresponding *ghost volume*. The *feature* is machinable, if the tool can machine the *feature* without the tool-part interference.

The conditions for the machinability of various *features* are described in forthcoming sections.

2.5.1 T-slot and T-slot Milling Cutter

Fig 2.14.a and Fig 2.14.b show a t-slot and a t-slot cutter respectively. Searching of a tool for a t-slot is done by matching body width of t-slot with tool body diameter and body thickness with the tool body thickness. The various conditions to be satisfied by different cases of machinable t-slots are discussed below. The terminology used in the discussion is explained in Fig 2.14.

Case 1. $t_d = 0$ and $n_w \geq tn_d$ and $n_t \leq tn_1$. An instance of this case is shown in Fig 2.15a.

Case 2. $t_d = 0$ and $n_w \geq ts_d$ and $n_t \leq tr_1$. An instance of this case is shown in Fig 2.15b.

Case 3. $t_d \neq 0$ and $n_w \geq tn_d$ and $(b_d - b_t) \leq tn_1$. Fig 2.15c shows an instance of this case and the following conditions are explained with respect to the Fig 2.115c and 2.14.

$(X_c - X_{left}) \geq (tn_d / 2)$ and $(X_{right} - X_c) \geq (tn_d / 2)$ for all slots above the t-slot, i.e. slots 1, 2, 3, and 4 in Fig 2.15c. X_{left} and X_{right} are left and right X-extents of a slot.

Case 4. $t_d \neq 0$ and $(b_d - b_t) > tn_1$ and $b_d \leq ta_1$. If $n_t \leq tn_1$ then $n_w \geq tn_d$ else $n_w \geq ts_d$. Fig 2.15d shows an instance of this case and the following conditions are explained with respect to

$(X_c - X_{left}) \geq (tn_d / 2)$ and $(X_{right} - X_c) \geq (tn_d / 2)$ for all slots above the t-slot such that $(Y_{top} - Y_c) \leq tn_1$, i.e. slots 3 and 4 in Fig 2.15d. X_{left} and X_{right} are left and right extents of the slot and Y_{bottom} and Y_{top} are bottom and top extents of the slot.

$(X_c - X_{left}) \geq (ts_d / 2)$ and $(X_{right} - X_c) \geq (ts_d / 2)$ for all slots, above t-slot, such that $(Y_{top} - Y_c) > tn_1$, i.e. slots 1 and 2 in Fig 2.15d.

2.5.2 Dovetail and a Dovetail Milling Cutter

Fig. 2.16.a and Fig 2.16.b show a dovetail and a dovetail cutter. A tool to machine a dovetail is found by matching its angle, α , with tool angle, α_t , and bottom width of dovetail with bottom dia of tool. The following discussion enumerates the various cases of machinable dovetails and the conditions which they satisfy. The terminology refers to Fig 2.16.

Case 1. $t_d = 0$ and $b_t \leq tb_t$. An instance of this case is shown in Fig 2.17a.

Case 2. $t_d \neq 0$ and $b_t = tb_t$ and $b_d \leq (tt_1 - ts_1/4)$. Fig 2.17b shows an instance of this case.

$(X_c - X_{left}) \geq (\text{tool dia at } Y_{top} / 2)$ and $(X_{right} - X_c) \geq (\text{tool dia at } Y_{top} / 2)$ for all slots above the dovetail. Tool dia at Y_{top} is $\text{minimum}\{ ts_d, tne_d + (ts_d - tne_d) * (Y_{top} - Y_{top \text{ of dovetail}}) / tn_1 \}$. If tool dia at $Y_{top} < t_w$ dia at top = t_w .

2.5.3 Reverse Dovetail and Reverse Dovetail Milling Cutter

Fig. 2.18.a and 2.18.b show a reverse dovetail and a reverse dovetail cutter. The tool selection and machinability checking are similar to the procedure followed for dovetails.

2.5.4 V-slot and V-slot Milling Cutter

Fig. 2.19a and Fig. 2.19b show a v-slot and a v-slot cutter. Tool is selected for a v-slot by matching its angle, α , with tool angle, α_t . The conditions for two cases of machinable v-slots are explained below. The terminology refers to Fig 2.19.

Case 1. $t_d = 0$ and $w \leq t_w$. An instance of this case is shown in Fig 2.20a.

Case 2. Fig 2.20b is an instance of this case. $t_d \neq 0$ and $w \leq t_w$ and $(t_d + h) < (t_{e_d} - t_{i_d}) / 2$. $(X_c - X_{left}) \geq (\text{tool width at } Y_{top})$ and $(X_{right} - X_c) \geq (\text{tool width at } Y_{top})$ for all slots above the v-slot, slots 1, 2, 3 and 4 in Fig 2.20b. X_{left} , X_{right} , Y_{bottom} and Y_{top} are X and Y extents of a slot. Tool width at Y_{top} is $\text{minimum}\{ w, (h + Y_{top} - Y_{top \text{ of v-slot}}) * (w / h) \}$.

2.5.5 Partial Dovetail and Dovetail Cutter

Fig 2.21a and Fig. 2.21b show a partial left dovetail and a partial right dovetail. A dove tail cutter is used to machine these partial features. The selection of tool is carried out by matching its angle, α , with that of the tool. A machinable partial dovetail should satisfy the following conditions. The terminology refers to Fig 2.21 and Fig 2.16.b.

$t \leq t_{b_t}$ and $(X_{right} - X_{left} + w) \geq t_{b_d}$ for the slot which is adjacent to partial dovetail and $(X_c - X_{left}) \geq \text{tool dia at } Y_{top} / 2$ for all slots above the partial left dovetail, i.e. slots 1 and 2 in Fig 2.21a, and $(X_{right} - X_c) \geq \text{tool dia at } Y_{top} / 2$ for all slots above the partial right dovetail, slots 1 and 2 in Fig 2.21b. Tool dia at $Y_{top} = \text{minimum}\{ t_{s_d}, t_{ne_d} + (t_{s_d} - t_{ne_d}) *$

2.5.6 Partial Reverse Dovetail and Reverse Dovetail Cutter

Fig 2.22a and fig 2.22b show a partial reverse left dovetail and partial reverse right dovetail. The tool selection is carried out by matching its angle, α , with that of the tool. A machinable partial reverse dovetail should satisfy the following conditions. The terminology of the following discussion refers to Fig 2.22 and Fig 2.18.b.

$t \leq tb_t$ and $(X_{right} - X_{left}) \geq tb_d$ for a slot which is adjacent to partial reverse dovetail and $(X_c - X_{left}) \geq \text{tool dia}$ at $Y_{top}/2$ for all slots above the partial reverse left dovetail and $(X_{right} - X_c) \geq \text{tool dia}$ at $Y_{top}/2$ for all slots above the partial reverse right dovetail. Tool dia at $Y_{top} = \text{minimum}\{ts_d, tne_d + (ts_d - tne_d) * (Y_{top} - Y_{top \text{ of reverse dovetail}}) / tn_1\}$.

2.5.7 Partial Reverse Dovetail and Single Angle Cutter

Fig 2.23a shows a partial reverse left dovetail (prldt), which cannot be machined by a reverse dovetail cutter, but can be machined by a single angle cutter shown in Fig 2.23b.

All the reverse partial dovetails which are not machinable by a reverse dovetail cutter are examined for the machinability, using single angle cutters.

The the single angle tool is selected by matching the angle α with that of the tool. Differnt cases of machinable prldts are discussed below.

Case 1. $w \leq t_w$ and $t_d = 0$, as shown in Fig 2.24a.

Case 2. $w \leq t_w$ and $t_d \neq 0$ and $b_d \leq (te_d - ti_d)/2$. Refer Fig 2.24b.

$(X_{right \text{ of prldt}} - X_{left \text{ of the slot}}) \geq \text{tool width at } Y_{top}$ of the slot for all slots above the prldt such that Y_{top} of the slot $\leq (Y_{bottom \text{ of the prldt}} + \text{tool height})$, i.e. slots 3 and

4shown in Fig 2.24b.

If the above condition is not satisfied for all slots then the right edges of the slots below prldt are checked. Let X'_{left} is the left extent of the slot upto which above condition is satisfied. Then the following condition should be satisfied for a machinable rpldt.

$(X_{right} \text{ of the slot} - X'_{left}) \leq \text{tool width at } Y_{bottom} \text{ of the slot}$ for all slots such that $((X_{left} \text{ of plrdt} - X'_{left}) * \cot \alpha + Y_{bottom} \text{ of plrdt} - Y_{bottom} \text{ of the slot}) \leq \text{tool height}$ i.e. slots 1 and 2 in Fig 2.24b.

2.5.8 Partial T-slot and T-slot Milling Cutter

Fig 2.25a and Fig 2.25b. show a partial left t-slot and partial right t-slot. The tool selection is done by matching the thickness of partial t-slot with that of the tool. A machinable partial t-slot should satisfy the following conditions. The terminology refers to Fig 2.25 and Fig 2.14b.

$(X_{right} - X_{left} + w) \geq tb_d$ for the slot which is adjacent to the partial reverse dovetail. $(X_c - X_{left}) \geq ts_d / 2$ for all slots above the partial left t-slot such that $(Y_{top} - Y_{top} \text{ of partial t-slot}) \geq \text{tool neck length}$ i.e. slot 3 in Fig 2.25a. $(X_c - X_{left}) \geq tn_d/2$ for all slots above the partial left t-slot such that $(Y_{top} - Y_{top} \text{ of partial t-slot}) < tn_1$, i.e. slots 1 and 2 in Fig 2.25a. $(X_{right} - X_c) \geq ts_d/2$ for all slots above the partial right t-slot and $(Y_{top} - Y_{top} \text{ of partial t-slot}) \geq tn_1$, i.e. slot 4 in Fig 2.25b. $(X_{right} - X_c) \geq tn_d/2$ for all slots above the partial right t-slot and $(Y_{top} - Y_{top} \text{ of partial t-slot}) < tn_1$, i.e. slots 1, 2 and 3 in Fig 2.25b.

2.6 MERGING PARTIAL FEATURES

Conditions for merging various partial features are given below.

2.6.1 Merging of Two Partial Dovetails

A partial left dovetail and a partial right dovetail are merged if they satisfy following conditions. Refer Fig 2.26

Thicknesses, t and angles, α of partial dovetails match. A tool is found such that tool angle, α_t , = angle of partial dovetail, α , and tool body thickness, $tb_t \geq$ thickness of partial dovetail, t . They are separated by an adjacent slot such that $(X_{\text{right}} - X_{\text{left}} + 2 * \text{width of partial dovetails}) = \text{tool bottom dia.}$

2.6.2 Merging of Two Partial Reverse Dovetails

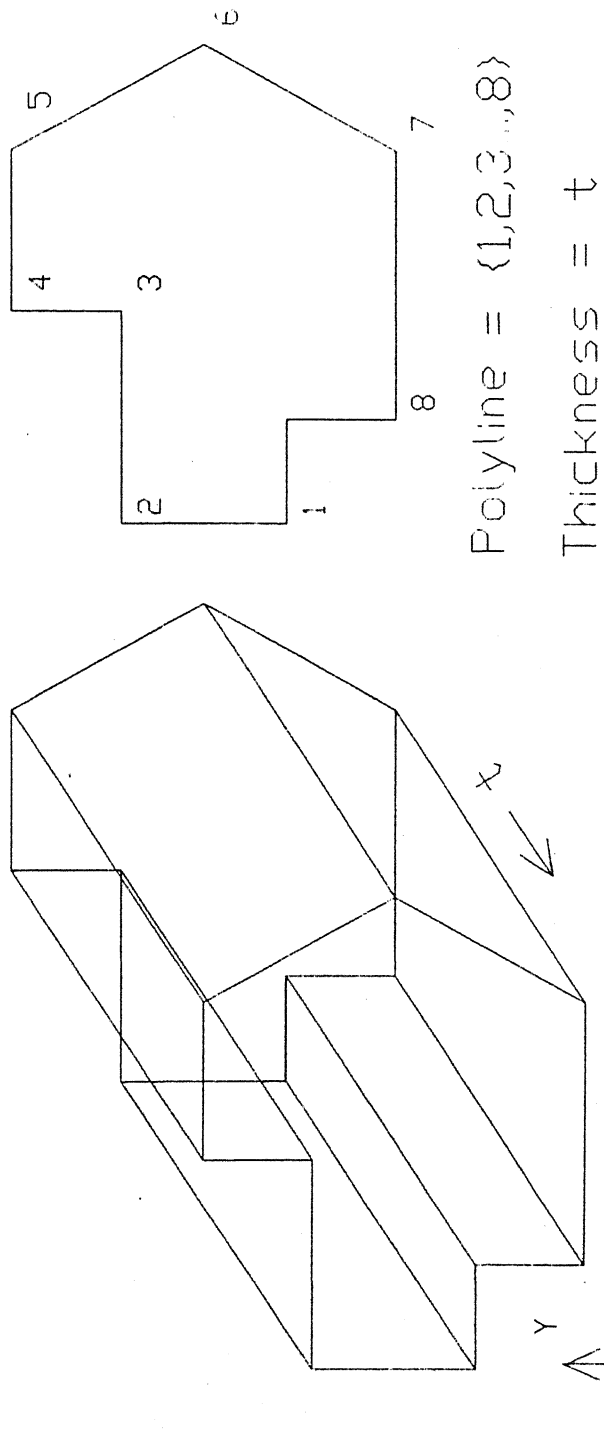
A partial reverse left dovetail and a partial reverse right dovetail are merged if they satisfy the following rules. Refer Fig 2.27.

Thicknesses, t and angles, α of partial reverse dovetails match. A tool is found such that tool angle, α_t , = angle of partial reverse dovetail, α , and tool body thickness, $tb_t \geq$ thickness of partial reverse dovetail, t . They are separated by an adjacent slot such that $(X_{\text{right}} - X_{\text{right}}) = \text{tool bottom dia.}$

2.6.3 Merging of Partial T-slots

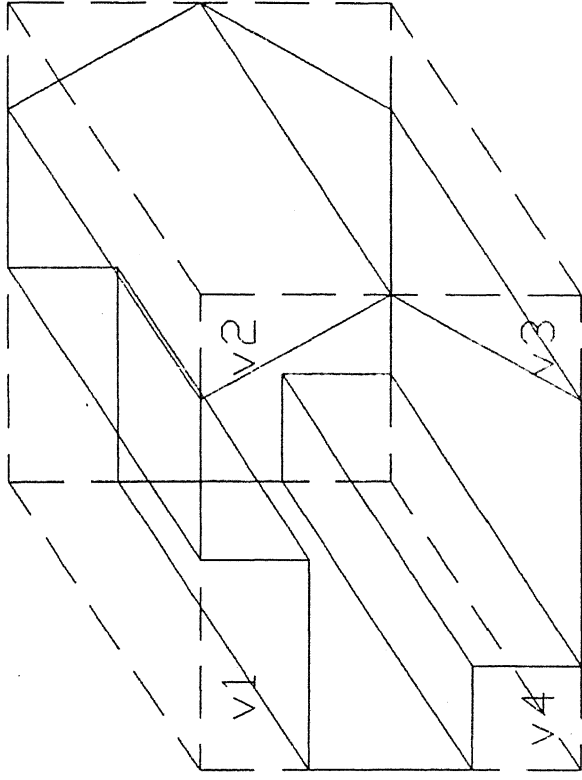
A partial left t-slot and a partial right t-slot are merged if they satisfy the following conditions. Refer Fig 2.28.

Thicknesses of partial t-slots match. A tool is available such that tool body thickness, $tb_t =$ thickness of partial t-slots, t . They are separated by an adjacent slot such that $(X_{\text{right}} - X_{\text{left}} + \text{width of partial left t-slot} + \text{width of partial right t-slot}) = tb_d$.



a. 2.5-D part b. AutoCAD model

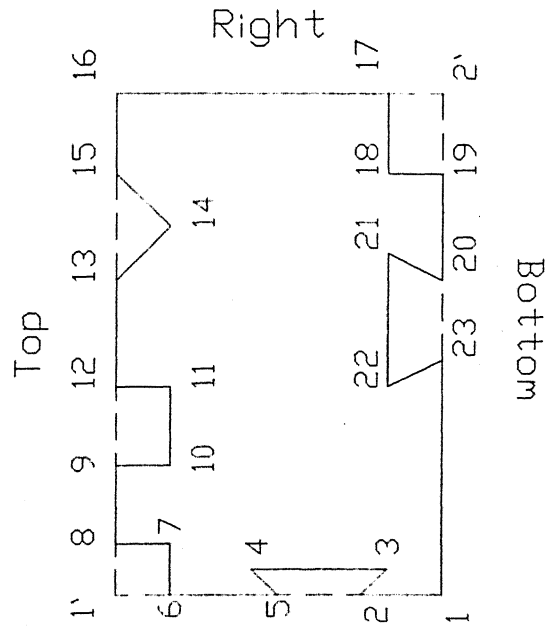
Fig 2.1 A 2.5-D part and its AutoCAD model



Cavity volume = $\{v1, v2, v3, v4\}$

Ghost volumes = $\{v1\}, \{v2\}, \{v3\}, \{v4\}$

Fig 2.2 Cavity volume and ghost volumes of the part shown in Fig 2.1a



Base Plane : 1 2 3 4 ... 23
 Enclosing Rectangle : 1 1' 16 2'
 Lower left corner : 1
 Deflection point from left to top : 8
 Deflection point from top to right : 16
 Deflection point from right to bottom : 19
 Deflection point from bottom to left : 1
 Left linked list : 1 2 3 4 5 6 7 8
 Top linked list : 8 9 10 11 12 13 14 15 16
 Right linked list : 16 17 18 19
 Bottom linked list : 19 20 21 22 23 1

Fig 2.3 Schematic representation of Steps 1, 2, 3 and 4 of feature recognition algorithm

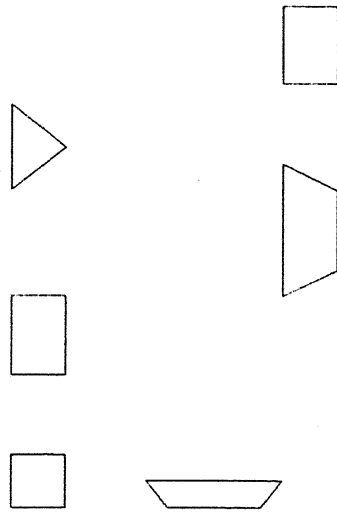
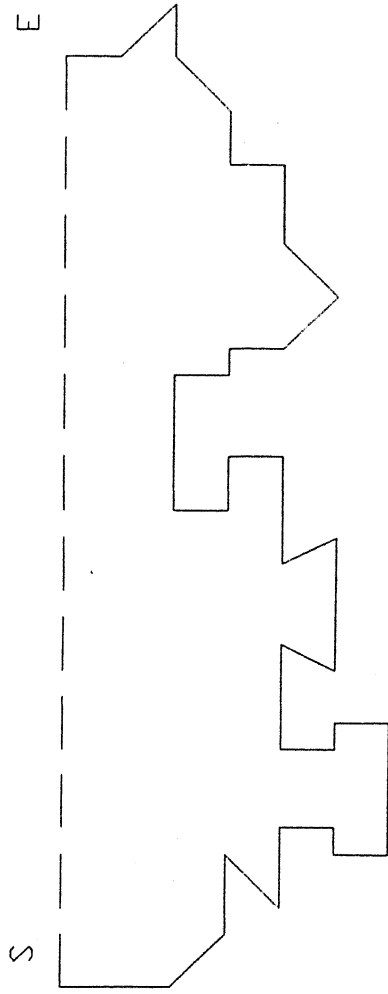
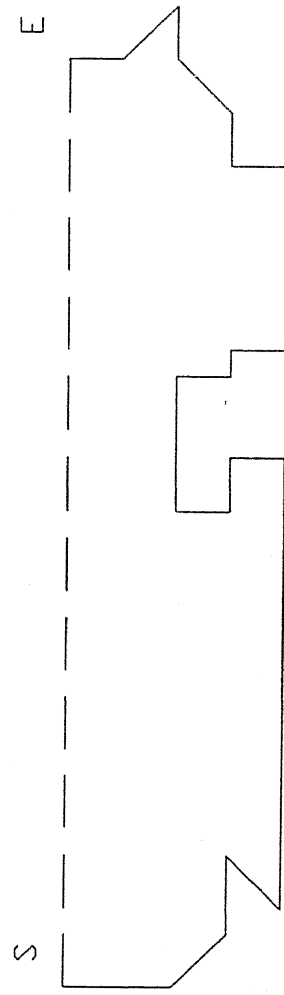


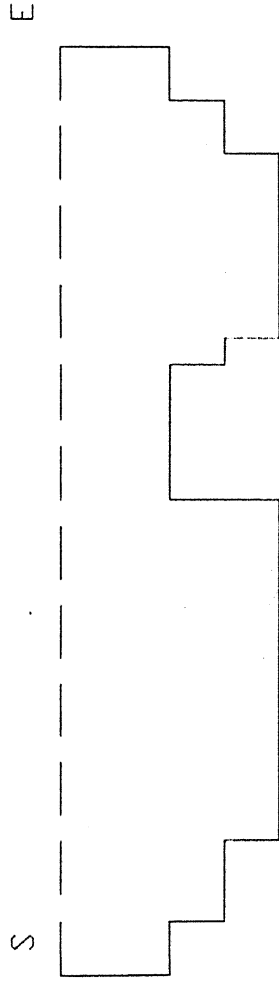
Fig 2.4 Polygons of ghost volumes corresponding to
to base plane in Fig 2.3



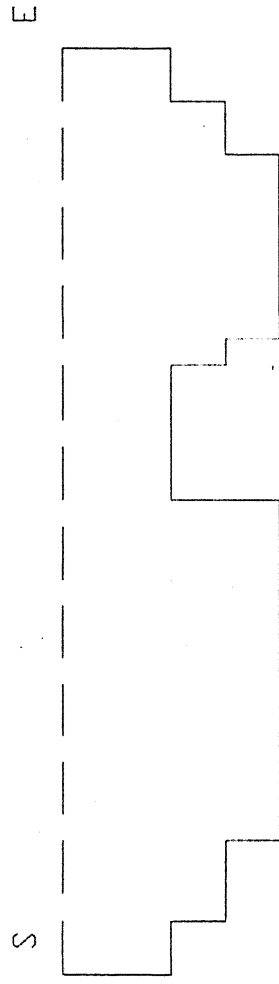
a. Ghost volume polygon before the extraction of features



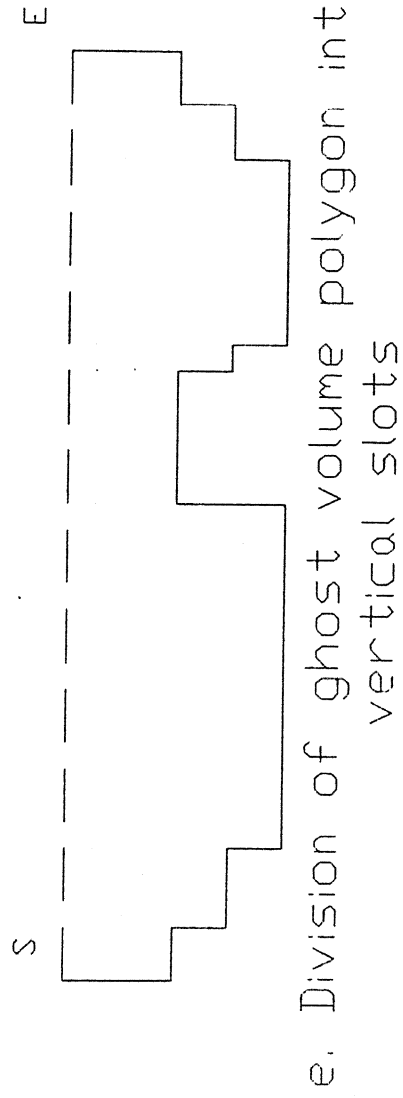
b. Ghost volume polygon after the extraction of complete features



c. Ghost volume polygon after the extraction of complete and partial features



d. Division of ghost volume polygon into horizontal slots



S : start point
E : end point

Fig 2.5 Schematic representation of Step 5 of feature recognition algorithm

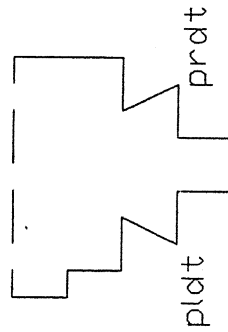


Fig 2.6 Mergable (matching) partial dovetails

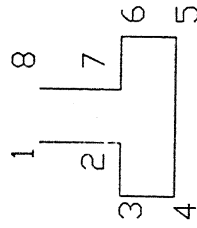


Fig 2.7 T-slot as a pattern of 8-point sequence

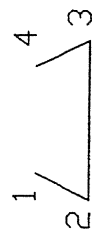


Fig 2.8 Dovetail as a pattern of 4-point sequence

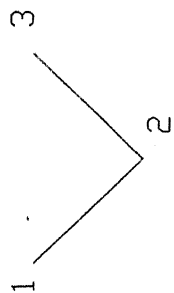


Fig 2.9 V-slot as a pattern of 3-point sequence

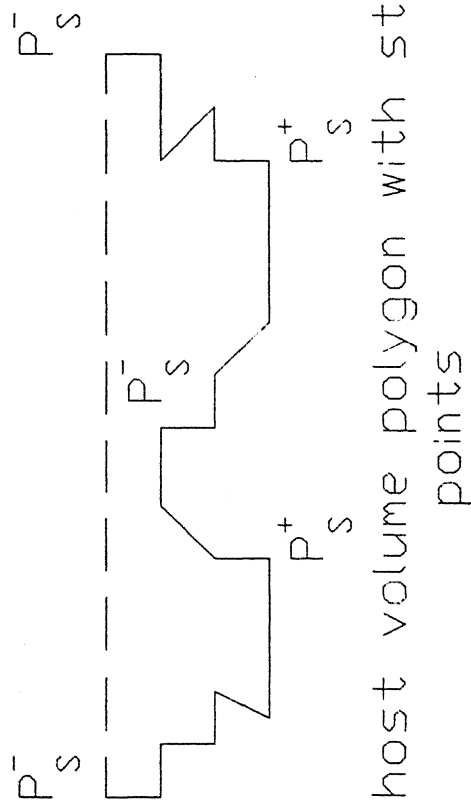


Fig 2.10 A ghost volume polygon with stationary points

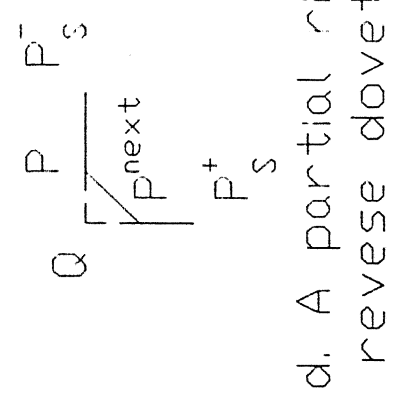
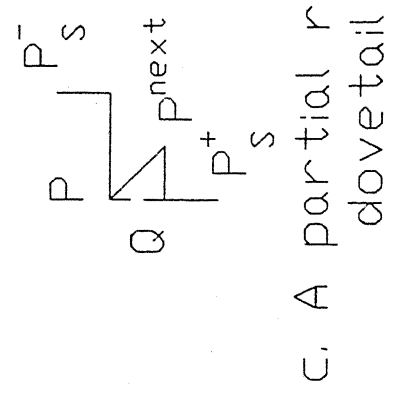
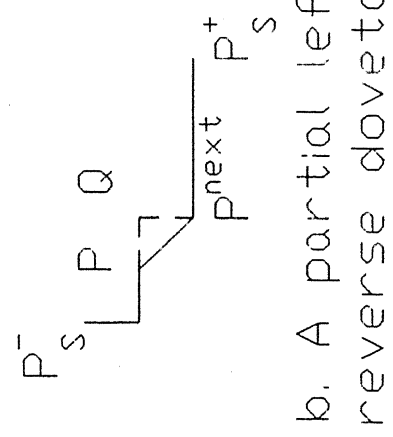
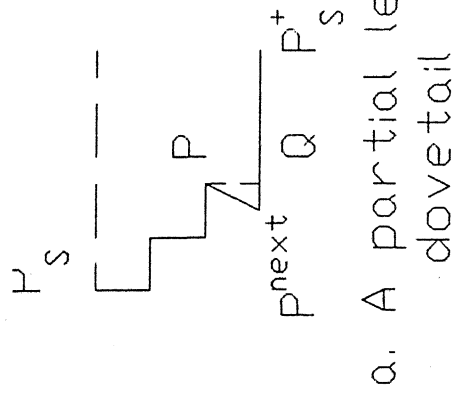


Fig 2.11 Schematic illustration of the extraction of partial dovetails

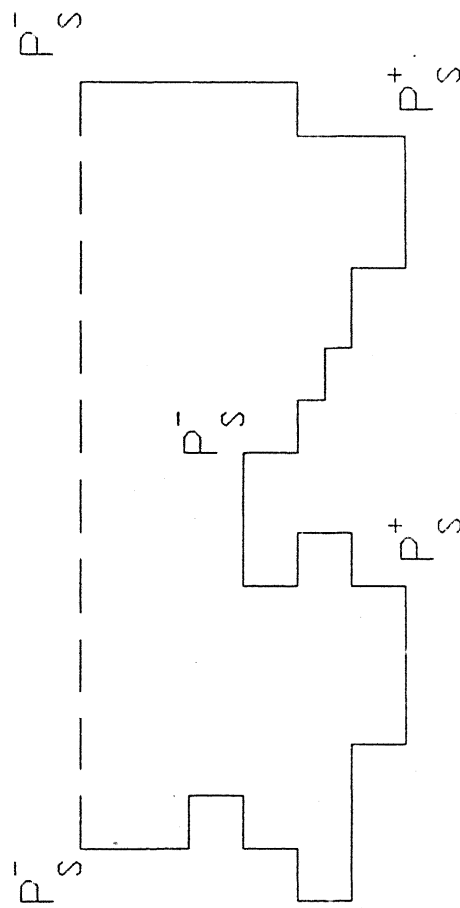
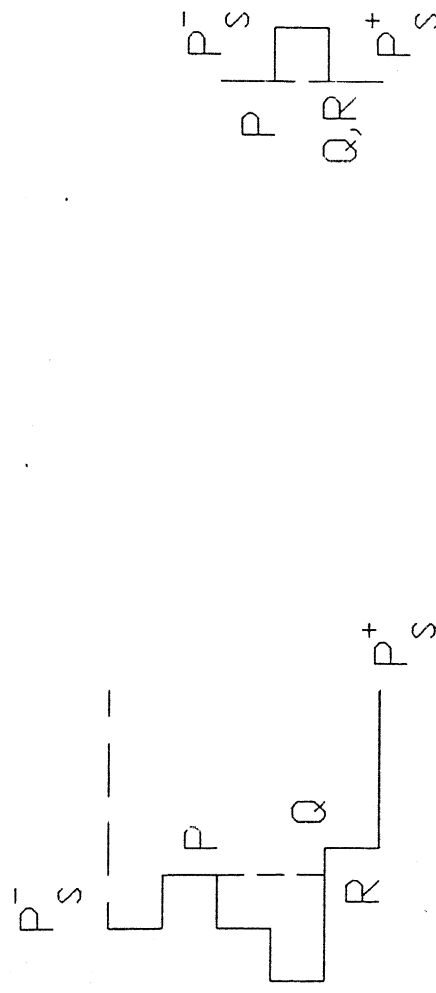
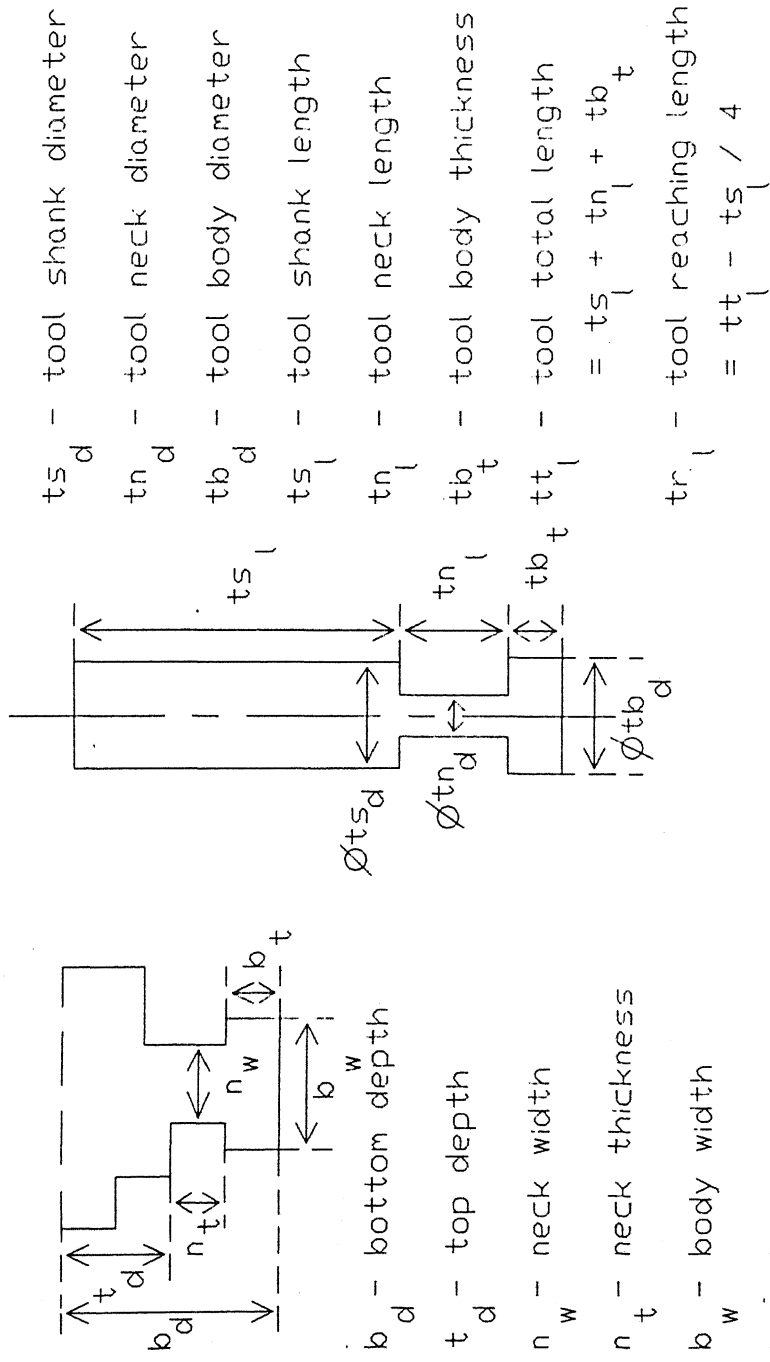


Fig 2.12 A ghost volume polygon with stationary points after the extraction of complete features and partial dovetails



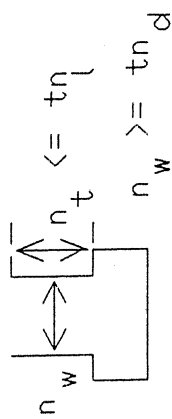
a. Partial left T-slots b. A partial right T-slot

Fig 2.13 Schematic illustration of the extraction of partial T-slots

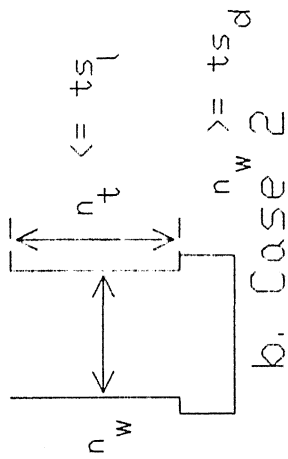


2.14a A T-slot and its geometry

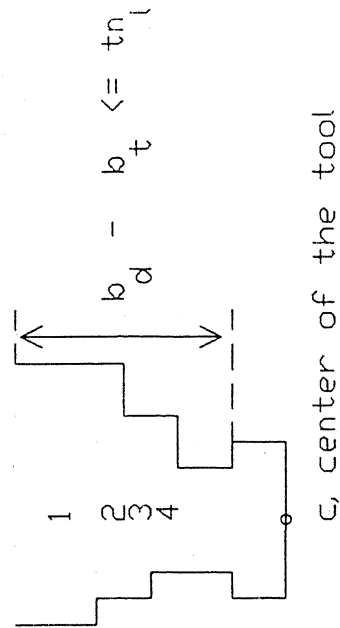
2.14b A T-slot milling cutter and its geometry



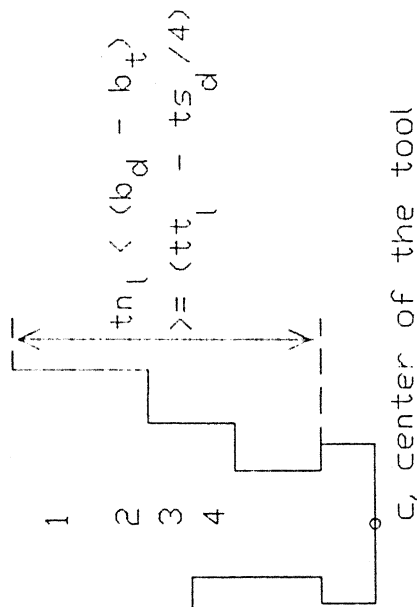
a. Case 1



b. Case 2

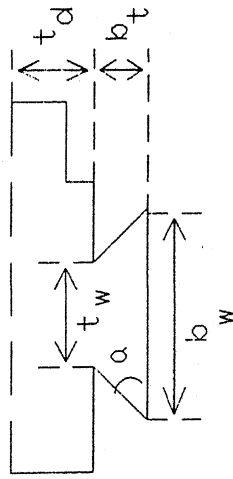


c. Case 3



d. Case 4

Fig 2.15 Machinable T-slots



t_d - top depth

b_t - body thickness

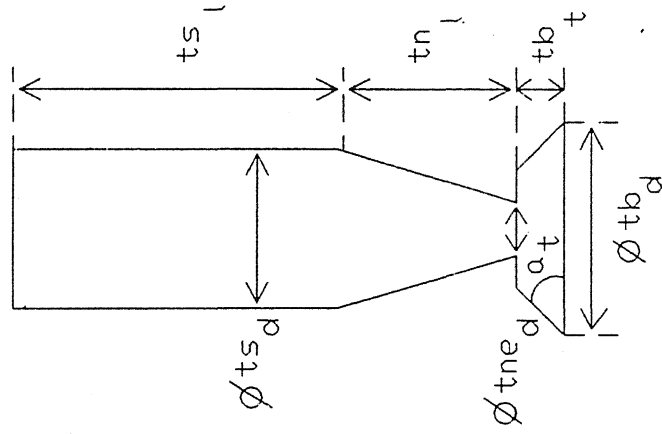
b_w - body width

t_w - top width

a - Dovetail angle

b_d - bottom depth

$$= t_d + b_t$$



ts_l - tool shank length

tn_l - tool neck length

tb_t - tool body thickness

ts_d - tool shank diameter

tne_d - tool neck end dia.

tb_d - tool body diameter

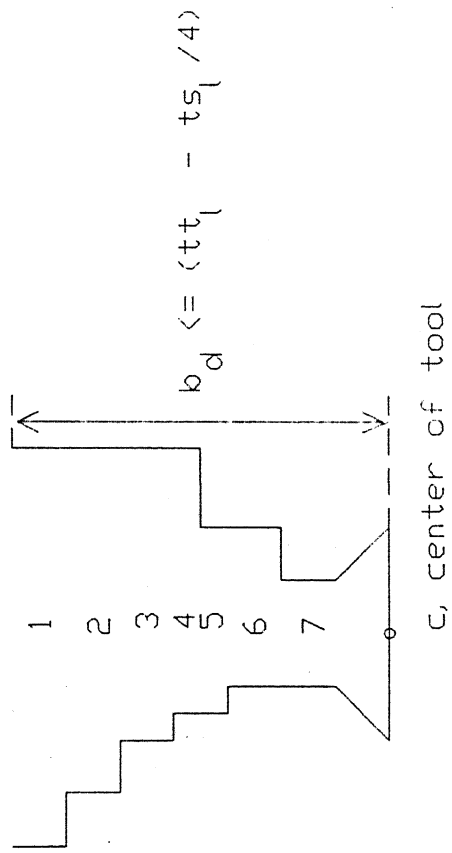
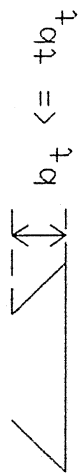
tt_l - tool total length
 $= ts_l + tn_l + tb_t$

tr_l - tool reaching length
 $= tt_l - ts_l / 4$

a - tool angle

Fig 2.16a A dovetail and its geometry

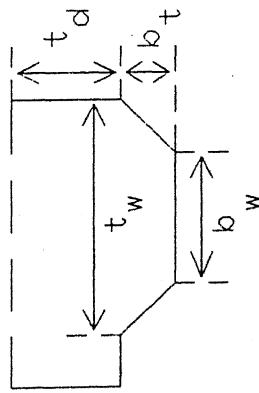
Fig 2.16b A dovetail milling cutter and its geometry



a. Case 1

b. Case 2

Fig 2.17 Machinable dovetails



t_d - top depth

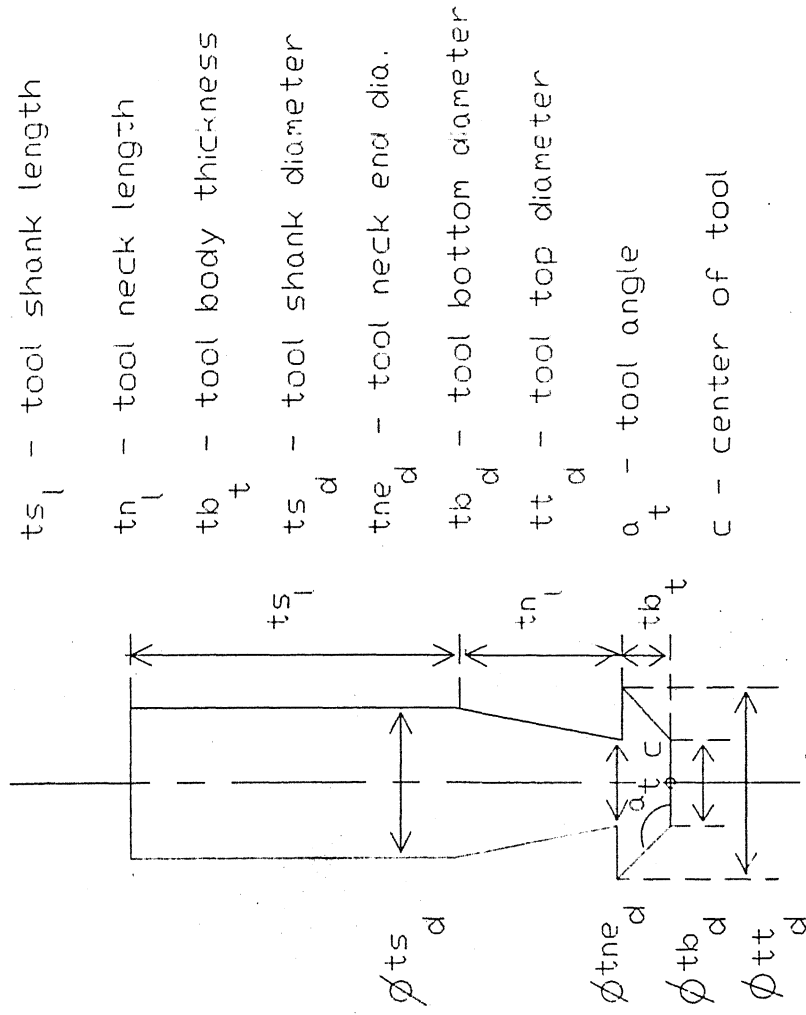
b_t - body thickness

b_w - bottom width

t_w - top width

b_d - bottom depth

Fig 2.18a A reverse dovetail and its geometry



ts_l - tool shank length

tn_l - tool neck length

tb_t - tool body thickness

ts_d - tool shank diameter

tne_d - tool neck end dia.

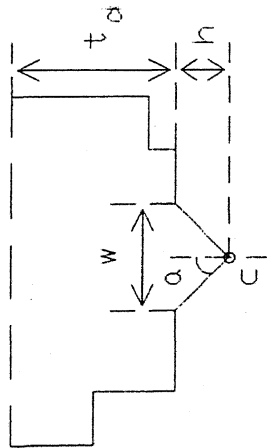
tb_d - tool bottom diameter

tt_d - tool top diameter

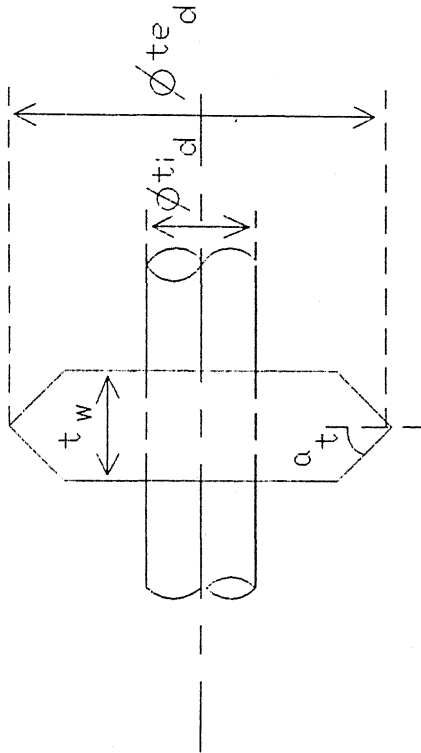
a_t - tool angle

c - center of tool

Fig 2.18b A reverse dovetail cutter and its geometry



w - width
 α - angle
 h - height
 t_d - top depth

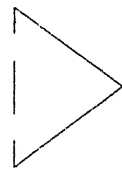


t_w - tool width
 α_t - tool angle
 ϕ_{ti_d} - tool internal dia.
 ϕ_{te_d} - tool external dia.

Fig 2.19a A V-slot and its geometry

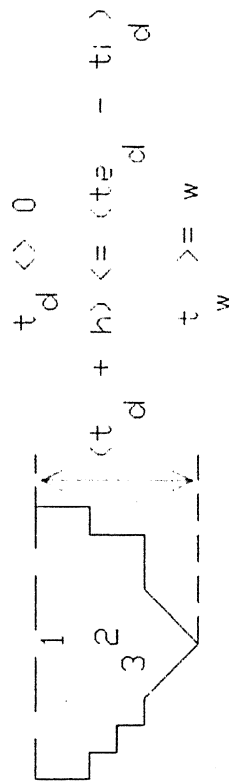
Fig 2.19b An equal angle milling cutter and its geometry

112202



$$t_d = 0$$

$$t_w \geq w$$



$$t_d < 0$$

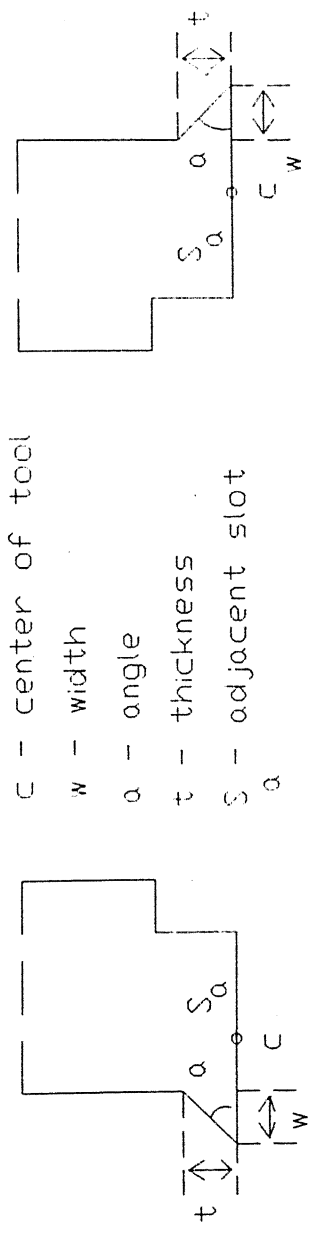
$$(t_d + h) \leq (t_d - t_i)$$

$$t_w \geq w$$

a. Case 1

b. Case 2

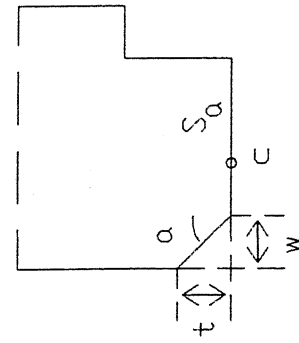
Fig 2.20 Machinable V-slots



c - center of tool
w - width
a - angle
t - thickness
 s_a - adjacent slot

Fig 2.21a A partial left dovetail and its geometry

Fig 2.21b A partial right dovetail and its geometry



c - center of tool
 w - width
 a - angle
 t - thickness
 S_a - adjacent slot

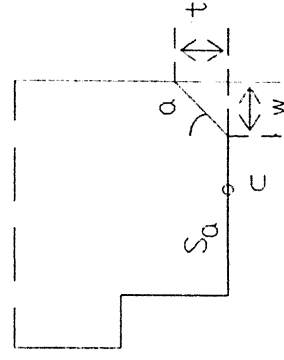
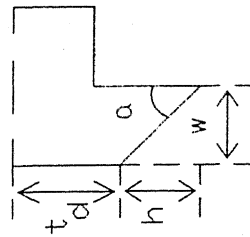


Fig 2.22a A partial left reverse dovetail and its geometry

Fig 2.22b A partial right reverse dovetail and its geometry



t_d - top depth

h - height

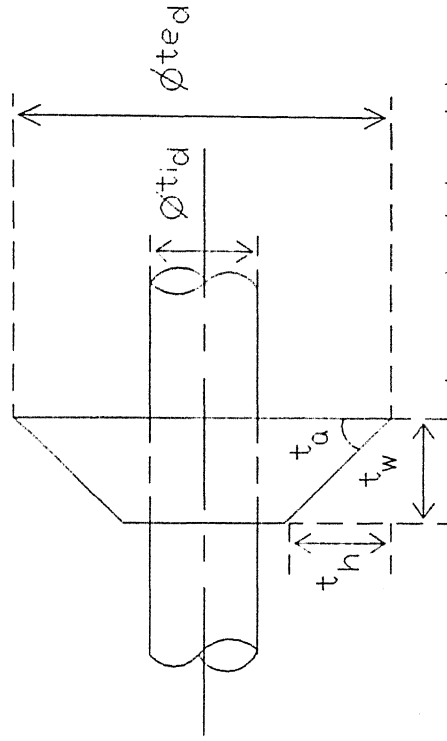
w - width

a - angle

b_d - bottom depth

$$= (t_d + h)$$

Fig 2.23a A partial reverse left dovetail and its geometry



t_h - tool height

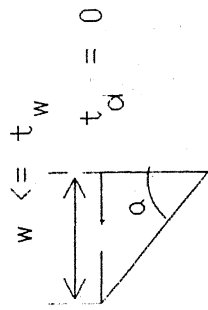
t_w - tool width

t_a - tool angle

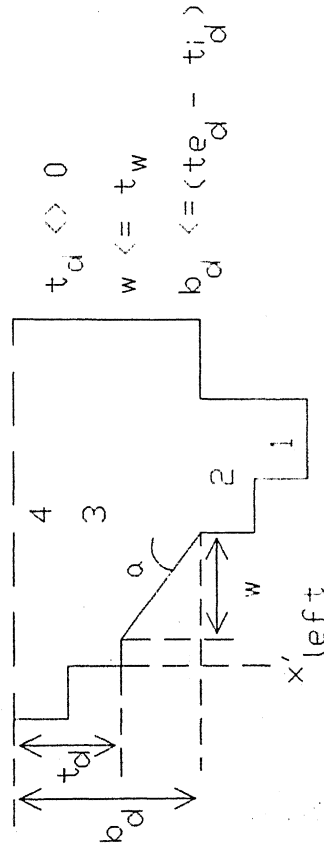
t_{e_d} - tool external dia

t_{i_d} - tool internal

Fig 2.23b A single angle milling cutter and its geometry



a. Case 1



b. Case 2

Fig 2.24 Machinable partial left reverse dovetails
(using single angle milling cutter)

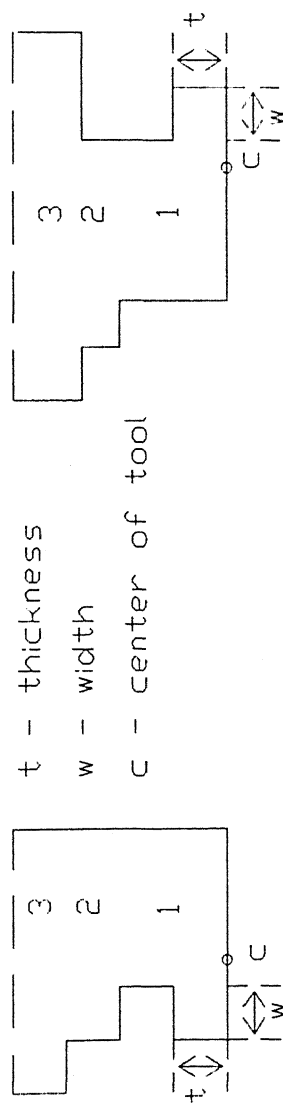


Fig 2.25a Partial left Fig 2.25b Partial right
T-slot and its geometry T-slot and its geometry

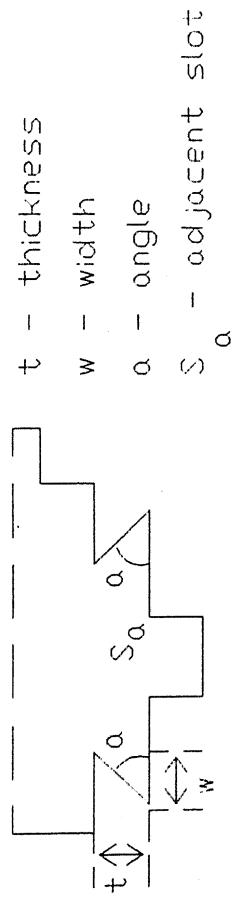


Fig 2.26 Matching partial dovetails and their geometry

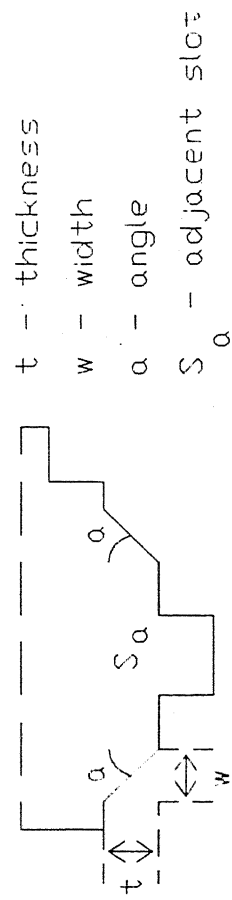


Fig 2.27 Matching partial reverse dovetails and their geometry

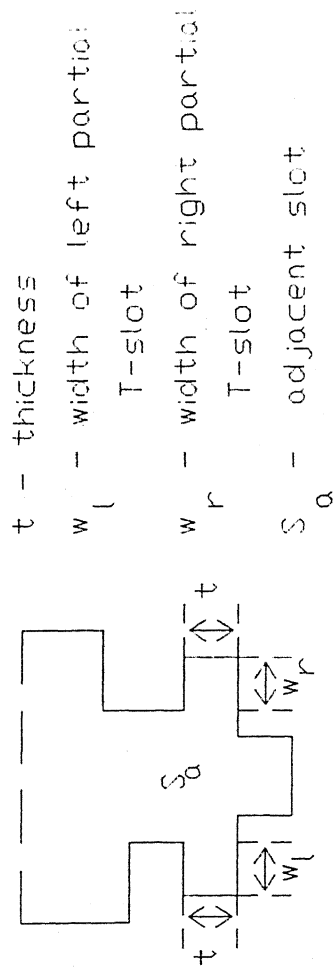


Fig 2.28 Matching partial T-slots and their geometry

Chapter III

FEATURE RECOGNITION OF 3-D PRISMATIC PARTS

In this chapter an algorithm for *feature recognition* of 3-d prismatic parts is discussed. The 3-D prismatic part is represented as an ensemble of boundary representations of the primitives that are constituting it. It may be noted that the CAD system allows a set of primitives and the solid portion of the 3-D part is considered as a union of some instances of these primitives excluding hole-type features. The hole-type features are represented in the boundary form without any explicit indication about their hollowness. One has to infer from the overall data whether a primitive is a hole feature or not. The present algorithm works for the objects, which can be designed using primitives such as cuboid, cylinder, wedge and partial cylinder. The types of primitives can be extended.

Since a 3-D part is designed as an ensemble of primitives, the unique and complete definition of primitives is very essential. The cuboids can be of two types - first type of cuboids have their sides parallel to principle axes and second type can have any other orientation. The former can be defined by specifying the extents of three principle sides, while the latter, cuboid2s, are represented by specifying the base plane, a rectangle, and a perpendicular extrusion vector in terms of its direction and magnitude. A wedge can be defined by a base plane, a triangle, and a perpendicular extrusion vector. A cylinder is defined by a base plane, a circle and a perpendicular extrusion vector. The circle has a center and a radius. A partial cylinder is defined by a base plane, an arc and a perpendicular extrusion vector. The arc has a center, a radius, a starting angle and an

ending angle. The primitives considered in the algorithm should have their extrusion vectors parallel to one of the principle axes. A right hand co-ordinate system and the right-hand thumb rule, to define the positive angle direction, are used in the algorithm.

In the present system, AutoCAD is used to model the part diagrams. Each of the primitives corresponds to an entity in AutoCAD. A cuboid and a wedge correspond to solids, a partial cylinder corresponds to an arc, with a certain thickness and a cylinder corresponds to a circle, with a certain thickness. Fig 3.1 shows the primitives used in designing a part.

The DXF file of a 3-D part contains the geometry of the primitives. AutoCAD uses world coordinate system (WCS) to define the geometry of primitives whose extrusion direction is parallel to Z-axis and uses entity coordinate system (ECS) to define the geometry of other entities. AutoCAD follows a conversion algorithm to define an ECS.

If $N = N_x i + N_y j + N_z k$ is the extrusion vector of an entity in WCS, then ECS is defined as follows:

Let W_x , W_y and W_z are the unit direction vectors along the principle axes of WCS.

If ($N_x < 1/64$) and ($N_y < 1/64$)

$$\text{then } A_x = (W_y \times N) \quad (3.0.1.a)$$

$$\text{else } A_x = (W_z \times N). \quad (3.0.1.b)$$

$$A_y = (N \times A_x). \quad (3.0.2.)$$

$$A_z = N \quad (3.0.3.)$$

On reducing to unit length A_x , A_y and A_z give the unit vectors in the X , Y and Z-directions of ECS. The origin in ECS is same as the origin in WCS.

The geometry of the entities is transformed from ECS to WCS, before the feature extraction is carried out.

3.1 METHODOLOGY

A hierarchical methodology is developed to find out the *cavity volume*. After the transformation of entities from ECS to WCS the *hole features* are found out. The primitives which are completely contained in other primitives are called *hole features*. The *hole features* are the *local ghost volumes* to be removed from the stock. Then the primitives other than cuboids are replaced by enclosing cuboids. The set difference of the primitives and enclosing cuboids gives the *local ghost volumes* to be removed from the stock. Now the 3-D part is reduced to an object which is an ensemble of cuboids, whose sides are parallel to three principle axis.

The stock is assumed to be the enclosing cuboid of the resultant 3-D part. The set difference of the stock and the reduced 3-D part gives a group of discrete *global ghost volumes*. The local and *global ghost volumes* combinedly make the *cavity volume* to be removed from the stock to manufacture the 3-D part. The *ghost volumes* require further processing to convert them into machinable features such as slots, dovetails, t-slots, etc..

3.2 ALGORITHM FOR THE EXTRACTION OF GHOST VOLUMES

Based on the above methodology the various steps involved in the algorithm are described below.

Step 1. (a) Read the DXF file of the part.

(b) Transform all the entities to WCS. The transformation is described in section 3.2.1. Each entity corresponds to a primitive.

Step 2. Extract the *hole features* by searching the primitives which are contained in other primitives. This is carried out as follows.

- (a) Replace the non-cuboid primitives with the enclosing cuboids, temporarily, as shown in Fig 3.2.
- (b) Sort the cuboids in the ascending order of Z_1 and Z_2 where Z_1 and Z_2 denote the Z-extents of a primitive. The order for Fig 3.2b is (1, 2, 3), (4, 5) and 6.
- (c) Carry out the adjacency analysis, which is explained in Section 3.2.2 to test the mergability of the cuboids, with same Z-extents. Adjacency analysis results in discrete groups of cuboids. Fig 3.3 shows the result of the adjacency analysis on part shown in Fig 3.2. It shows three diagrams, each of which shows the discrete groups of solids of the same Z-extents. Find out the cuboids which lie in one or more of other cuboids and classify the corresponding primitives as *hole features* and they are deleted from the cuboid list. In Fig 3.3a cuboid 2 lies in cuboid 1, so it is recognized as a cuboid enclosing a *hole feature*.

Merge the remaining cuboids, at different Z-extents. The merging of a group of adjoining solids of same Z-extents is explained in Section 3.2.3. This results in a 3-D part as an ensemble of merged solids.

- (d) Merge the solids along Z-direction, to find out the *hole features* as shown in Fig 3.4. Assuming the S_1 and S_2 are the merged solids resulting from Step (c) and they are represented as $S_1 = (B_1, Z_{11}, Z_{12})$ and $S_2 = (B_2, Z_{21}, Z_{22})$, where B represents 2-D boundary of the base plane and Z_1, Z_2 represent Z-extents of the solid. The merging results in two solids $(B_1 - B_2, Z_{11}, Z_{12})$ and (B_2, Z_{11}, Z_{22}) . The condition for

merging S_1 and S_2 is $Z_{12}=Z_{21}$. Section 3.2.4. explains different cases of merging along Z-axis.

- (e) Search for the solids which are contained in other solids. If a solid is contained in other solids then the corresponding primitives are hole features. Classify the hole features as blind or through. This is explained in Section 3.2.5.
- (f) Repeat the Steps 2.(b) to 2.(e) by rotating the part around X-axis by +90 degrees to find out the holes parallel to Y-axis and by rotating the part around Y-axis by -90 degrees to find out the features parallel to X-axis
- (g) Enclose the non-hole features by the cuboids permanently and find out the set differences of the enclosed cuboids and primitives to give the local ghost volumes.

Step 3. After removing the hole features and the local ghost volumes, the 3-D part consists of non-hollow cuboids only.

- (a) Find out the enclosing cuboid of the resultant 3-D part. The enclosing cuboid is assumed to be the stock, as shown in Fig 3.2b.
- (b) Divide the 3-D part into a series of 2.5-D parts. Assuming the division is done along Z-axis, scan from minimum Z in the positive direction and generate a 2.5-D part at every variation in the X-Y section of the part. Each 2.5-D part is a base plane, with a certain extrusion. The division of the part in Fig 3.6a is shown in Fig 3.6c.
- (c) Divide the stock cuboid into the same number of 2.5-D stocks, each of which will be an extrusion of a

rectangle with a certain extrusion.

- (d) Find out the set difference of the 2.5-D stocks and corresponding 2.5-D part. The procedure to find out the set difference is explained in Section 3.2.5. These set differences are called *global ghost volumes*.
- (e) Find out the tool approachable faces of each *ghost volume*. This is explained in Section 3.2.6.

Step 4. Process the local *ghost volumes*, excluding the hole features, and global *ghost volumes* in to machinable features like slots, t-slots and dovetails etc.. The cylindrical holes do not require any processing as each of them corresponds to a drilling operation. The non-cylindrical holes are merged if they are adjoining.

Though the above described algorithm is supposed to be a generalised one, the implementation is done only for the 3-D parts which can be modeled with cuboids, wedges and partial cylinders, describing non-hole features, and cylinders as hole features. Even with this relaxation the task of a blind / through hole recognition is not a trivial one. The low level algorithms for the various tasks involved in the algorithm are discussed below.

3.2.1 Transformation of the Primitives to WCS

As already discussed, the DXF output from AutoCAD needs a processing to convert the geometry of all primitives, which are described in ECS, to WCS by the reverse transformation of conversion algorithm (CA). Since the CA uses rotation matrix in its transformation and the inverse of a rotation matrix is a

transpose of the matrix, the reverse transformation is not a difficult task. The transformation for the primitives whose extrusion direction is parallel to X and Y-axes is discussed below.

3.2.1.1 Transformation of Cubiod2, Wedge and Cylinder

There are four possible extrusion directions for a primitive. They are $+i$, $-i$, $+j$ and $-j$. Each of the extrusion directions result in different ECS.

Case 1 : Fig 3.7 shows a cylinder with an extrusion direction, $N = i$. The CA derives an ECS, A_x , A_y and A_z as follows:

From equations 3.0

$$A_x = (W_z \times N) = (k \times i) = j = W_y$$

$$A_y = (N \times A_x) = (i \times j) = k = W_z$$

$$A_z = N = i = W_x$$

Hence if x' , y' and z' are the coordinates of a point in ECS the coordinates of the point in WCS, x , y and z are given as

$$x = z'$$

$$y = x'$$

$$z = y'$$

The above equations are used in transforming the four vertices of the base plane of a cuboid, the three vertices of the base plane of a wedge and the center of the base plane of a cylinder.

Case 2 : Fig 3.8 shows a cylinder the extrusion direction, $N = -i$. The CA derives an ECS, A_x , A_y and A_z as follows:

From equations 3.0.

$$A_x = (W_z \times N) = (k \times -i) = -j = -W_y$$

$$A_y = (N \times A_x) = (-i \times -j) = k = W_z$$

$$A_z = N = -i = -W_x$$

Hence if x' , y' and z' are the coordinates of a point in ECS the coordinates of the point in WCS, x , y and z are given as

$$x = -z'$$

$$y = x'$$

$$z = -y'$$

The above equations are used in transforming the four vertices of the base plane of a cuboid, the three vertices of the base plane of a wedge and the center of the base plane of a cylinder.

Case 3 : Fig 3.9 shows a wedge with an extrusion direction, $N = j$. The ACA derives an ECS, A_x , A_y and A_z as follows:

From equations 3.0

$$A_x = (W_z \times N) = (k \times j) = -i = -W_x$$

$$A_y = (N \times A_x) = (j \times -i) = k = W_z$$

$$A_z = N = j = W_y$$

Hence if x' , y' and z' are the coordinates of a point in ECS the coordinates of the point in WCS, x , y and z are given as

$$x = -x'$$

$$y = z'$$

$$z = y'$$

The above equations are used in transforming the four vertices of the base plane of a cuboid, the three vertices of the base plane of a wedge and the center of the base plane a cylinder.

Case 4 : Fig 3.10 shows a cuboid2 with an extrusion direction, $N = -j$. The ACA derives an ECS, A_x , A_y and A_z as follows:

From equations 3.0

$$A_x = (W_z \times N) = (k \times -j) = i = W_x$$

$$A_y = (N \times A_x) = (-j \times i) = k = W_z$$

$$A_z = N = -j = -W_y$$

Hence if x' , y' and z' are the coordinates of a point in ECS the coordinates of the point in WCS, x , y and z are given as

$$x = x'$$

$$y = -z'$$

$$z = y'$$

The above equations are used in transforming the four vertices of the base plane of a cuboid, the three vertices of the base plane of a wedge and the center of the base plane of a cylinder.

3.2.1.2 Transformation of Partial Cylinder Primitives

Similar to the four cases of cuboid2s, wedges and cylinders partial cylinders are also four types depending on the extrusion direction. The transformation of the center of the base plane is similar to the transformations defined in the Section 3.2.1.2. The transformation of the start and end angles of the base plane can be explained as follows.

Case 1 : Fig 3.11 and Fig 3.12 show partial cylinders with extrusion directions, $N = i$ and $N = -i$ respectively. In ECS the start angle, s' and end angle, e' are defined around A_z with respect to A_y . In WCS the start angle, s and end angle, e are expressed around the X-axis with respect to Z-axis. The angles s and e are given by the following equations.

If $N = i$ then

$$s = (s' + 270) \bmod 360.$$

$$e = (e' + 270) \bmod 360.$$

If $N = -i$ then

$$s = (360 - (e' - 90)) \bmod 360.$$

$$e = (360 - (s' - 90)) \bmod 360.$$

Case 2 : Fig 3.13 and Fig 3.14 show partial cylinders with

extrusion direction S , $N = j$ and $N = -j$. In ECS the start angle, s' and end angle, e' are defined around A_z with respect to A_x . In WCS the start angle, s and end angle, e are expressed around the Y-axis with respect to Z-axis. The angles s and e are given by the following equations.

If $N = j$ then

$$s = (s' + 270) \bmod 360.$$

$$e = (e' + 270) \bmod 360.$$

If $N = -j$ then

$$s = (360 - (e' - 90)) \bmod 360.$$

$$e = (360 - (s' - 90)) \bmod 360.$$

3.2.2 Adjoining Analysis

Fig 3.15 shows six cuboids of same Z-extents. Before merging, the cuboids are divided into discrete groups such that in a group every cuboid has at least one adjoining cuboid. This is called adjoining analysis. A cuboid can be expressed as a base plane, a rectangle in X-Y plane, with an extrusion in Z-direction. Since the extrusion is same for all cuboids, it is adequate to consider the base planes of the cuboids, rectangles, in the adjacency analysis, which is described as follows:

Step 1. Sort the rectangles in the increasing order of X_1 .

The resulting order is 1, 2, 3, 4, 5, and 6.

Step 2. Scan the sorted list of rectangles to check X-mergability of a rectangle. The X-mergability is defined as follows.

-If (X_2 of the predecessor $\geq X_1$ of the rectangle)
then the rectangle is X-mergable with its predecessor.

Start a new group with every rectangle which is not X-mergable with its predecessor.

This results in two groups, (1, 2, 3) and (4, 5, 6), of rectangles. Now take up each group at a time and apply steps 3 and 4.

Step 3. Sort the rectangles in the increasing Y_1 order. The order for group (1, 2, 3) is 3, 1, and 2.

Step 4. Scan the sorted list of rectangles, to find out the Y-mergability of a rectangle, which is defined by the following rule.

-If (Y_2 of the predecessor $\geq Y_1$ of the rectangle)
then the rectangle is Y-mergable with its predecessor

Start a new group with every rectangle which is not Y-mergable with its predecessor.

This results in two groups (3) and (2, 1). Similarly the second group, (4, 5, 6) results in two groups (5) and (6,4).

The four groups (2, 1) , (3), (5) and (6,4) are the discrete groups of cuboids and the cuboids in each group can be merged.

3.2.3 Merging of a Group of Cuboids with same Z-extents

The next task after adjoining analysis is the merging of a group of cuboids, (1, 2, 3, 4), with same Z-extents. Fig 3.16 shows a group of cuboids to be merged. The merging algorithm, which is described here, merges the solids sequentially. i.e. In the group (1, 2, 3, 4), 1 and 2 are merged ,the resultant solid is merged with 3 and the resultant solid is merged with 4. Since the solids are of same thickness, it is adequate to consider the merging of their 2-D boundaries. The merging of two adjoining solids, 1 and 2, is described below.

Step 1. Define the base planes of the two solids, two

polygons, as counter clock-wise lists of vertices, as shown in Fig 3.17a and Fig 3.17b..

Step 2. Find out the intersection points of the two polygons and add the intersection points to the vertex lists of polygons, as shown in Fig 3.17c

Step 3. Tag each intersection vertex and establish a bidirectional link between the two polygons for each intersection vertex.

Step 4. To process the merged polygon, start at the first vertex of the first list and traverse in the forward direction.

(a) Follow the first vertex list until an intersection is encountered. Using the cross link jump to the second vertex list.

(b) Follow the second vertex list until an intersection is encountered and jump back to the first vertex list.

(c) Repeat the steps (a) and (b) until the start point is reached.

The path of traversal gives the base plane of merged solid. In Fig 3.17a the path of traversal is 1-2-3-1'-2'-3'-4'-4-1.

3.2.4. Merging of Two Solids along Z-axis

Depending on the relationship between the two solids, $S_1 = (B_1, Z_{11}, Z_{12})$ and $S_2 = (B_2, Z_{21}, Z_{22})$ the following cases can be enumerated.

Case 1. S_1 contains S_2 as shown in fig 3.18a. The merging results in two solids $S_3 = (B_1 - B_2, Z_{11})$ and $S_4 = (B_2, Z_{11}, Z_{22})$. In S_3 , B_2 is the hallow or hole portion and $B_1 - B_2$ is

Case 2. S_2 contains S_1 as shown in fig 3.18b. The merging results in two solids $S_3 = (B_1, Z_{11}, Z_{22})$ and $S_4 = (B_2 - B_1, Z_{21}, Z_{22})$.

Case 3. S_1 and S_2 contain each other as shown in fig 3.18c. The merging results in one solid $S_3 = (B_1, Z_{11}, Z_{22})$.

Case 4. S_1 and S_2 are disjoint, as shown in fig 3.18d. In this case the solids do not require any processing.

Case 5. S_1 and S_2 are intersecting each other, as shown in fig 3.18e. The merging involves finding the intersection area of the base planes of solids and it results in three solids, $S_1 = (B_1 - (B_1 \cap B_2), Z_{11}, Z_{12})$, $S_2 = (B_2 - (B_1 \cap B_2), Z_{21}, Z_{22})$, and $S_3 = ((B_1 \cap B_2), Z_{11}, Z_{22})$.

3.2.5 Finding Through and Blind Holes

After extracting the hole features, and merging the solids according to Section 3.2.3 and 3.2.4 the following rule can be used to find out whether each hole is a blind hole or a trough hole.

-If (a hole is completely contained in the solid portion of the merged volume)

then

If (extrusion of the hole = extrusion of the solid)

then it is a through hole,

else it is a blind hole.

-If (a hole is partly contained in the solid portion and partly contained in the hole portion of the merged volume)

then

the it is a blind hole.

3.2.6 Finding the Set Difference of Two Solids

Fig 3.19a and Fig 3.19b show two solids, a 2.5-D blank and a 2.5-D part, whose set difference is to be found out. Since the two solids are of same Z-extents, the difference between the base planes of the solids gives the base plane of the difference solid. To find out the set difference of the base planes, Weiler-Artherton algorithm is employed. To explain the algorithm the base planes of stock and part are designated as stock polygon and part polygon. Fig 3.19c and Fig 3.19d show the polygons corresponding to the solids in Fig 3.19a and Fig 3.19b.

- Step 1. Define the blank and part polygons as the doubly linked lists of the vertices, as shown in Fig 3.19e. Anti clock-wise direction is positive.
- Step 2. Find the intersection points of the two polygons and add them to the vertex lists, as shown in Fig 3.19f.
- Step 3. Tag the intersection points denoting the nature of the intersection point. i.e. whether an entering intersection or leaving intersection. The nature of the intersection is defined for the vertices of part polygon with respect to stock polygon. In Fig 3.19d 3' and 6' are leaving intersection vertices and 1' and 5' are entering intersection vertices.
- Step 4. For all leaving intersection vertices repeat the steps from (a) to (d).
 - (a). Start from the leaving intersection vertex and traverse the part polygon in the clock-wise direction.
 - (b) When an intersection point is encountered, switch to the stock polygon and traverse in the anti clock-wise direction.

- (c) Switch back to the part polygon, when an intersection point is encountered and traverse in the clock_wise direction.
- (d) Repeat the steps (b) and (c) until the start point is reached. The path of the total traversal gives a discrete areas of the set difference of two polygons.

Applying the above-algorithm to Fig 3.19a and Fig 3.19b results in the the set difference of two solids as two discrete ares as shown in Fig. 3.19g. The set difference volume is the linear sweep of the areas.

3.2.7 Tool Accessibility Analysis

Since the *ghost volumes* describe the volume to be removed from the stock, the recognition of faces which are accessible to tools is very important.

The tool accessible faces of a global *ghost volume* and local *ghost volumes* other holes are are found out by comparing the boundaries of its base plane with the boundary of the base plane of stock, a rectangle. The coincident boundary corresponds to the tool accessible faces. Fig 3.20a shows a 2.5-D stock, S and two *ghost volumes*, G_1 and G_2 . Fig 3.20b shows the base planes of the stock, S_b and *ghost volumes*, G_{b1} and G_{b2} The boundaries 1-2-3 and 4-5-6 of G_{b1} and G_{b2} coincide with S_b respectively. Hence the corresponding faces 1,2 and 3,4 are the tool approachable faces. The faces which are parallel to X-Y plane and at $Z = Z_1$ of the stock and $Z = Z_2$ of the stock are also tool accessible. This rule can be extrapolated for faces parallel to Y-Z and Z-X planes also.

Tool accessibility for hole features whose extrusion is

parallel to Z-axis is found out by the following rules.

-If the hole is a through hole

then it is accessible from both the sides, (Fig 3.21a)

Else

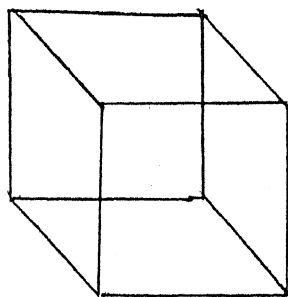
If the base plane is at $Z = Z_1$ of the stock

then it is accessible from base plane only,

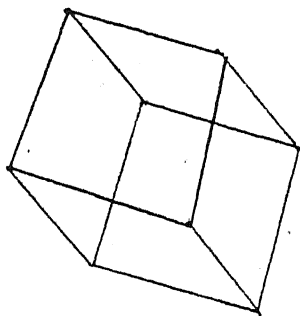
(Fig 3.21b)

Else it is accessible from the plane at $Z = Z_2$ of the stock. (Fig 3.21c)

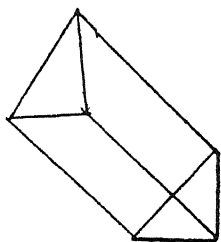
The logic can be extended to the hole features parallel to other sides.



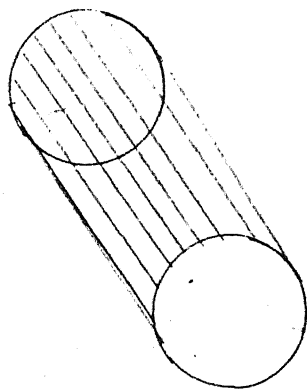
a. Cuboid



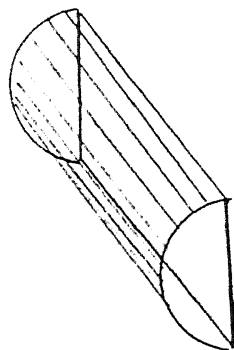
b. Cuboid2



c. Wedge



d. Cylinder



e. Partial cylinder

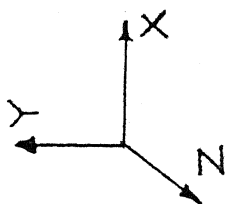


Fig 3.1 Primitives

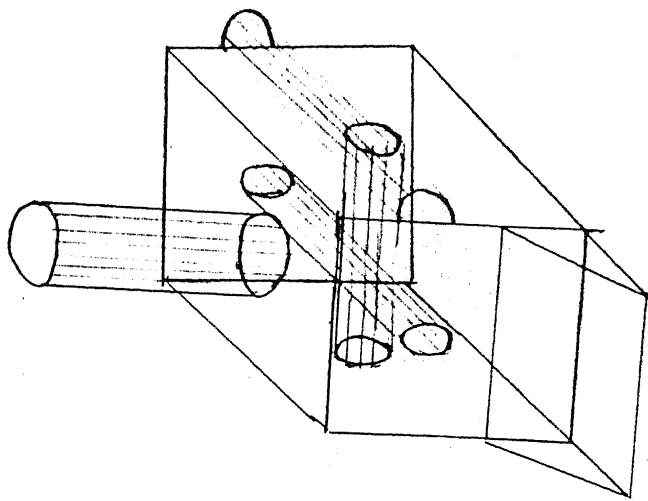


Fig 3.2a A 3-D part

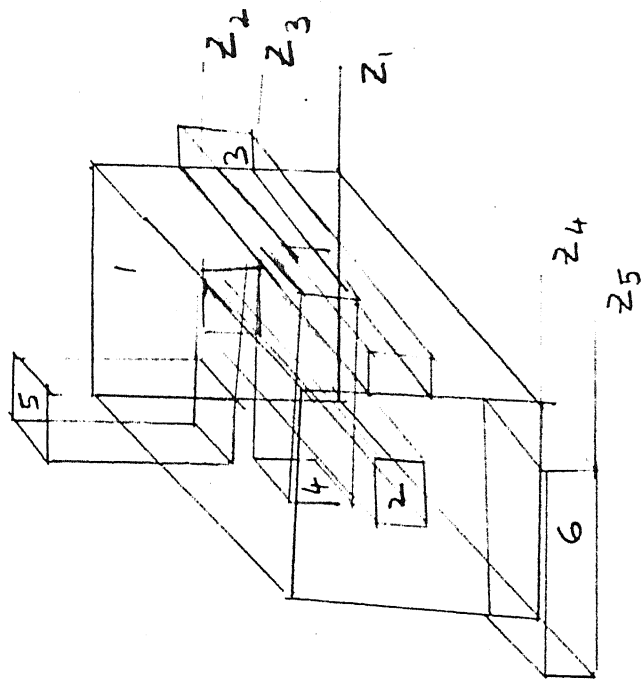
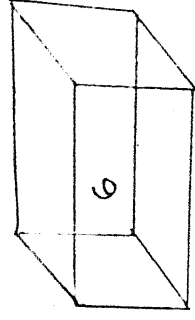
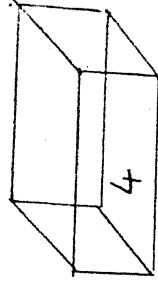
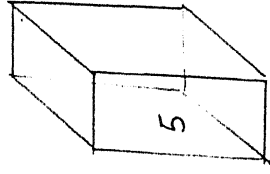
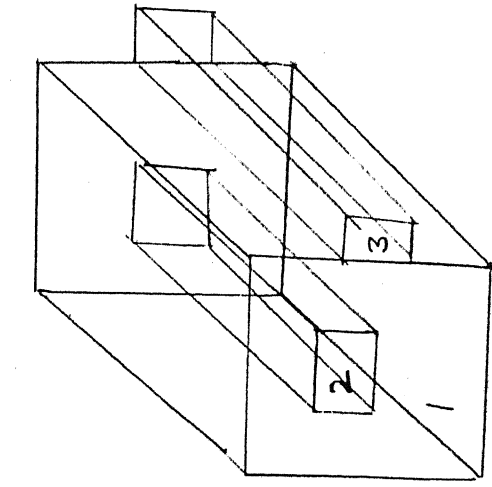


Fig 3.2b The 3-D part after enclosing the non-cuboid by culoids



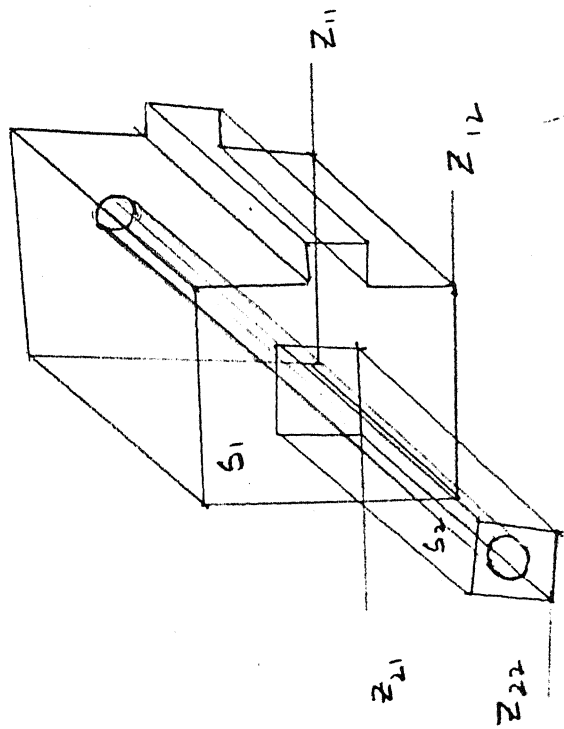
Cuboids with Z -extents
 Z_1 and Z_4

b. Cuboids with

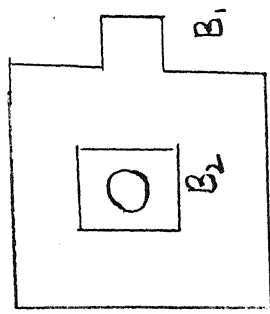
Z -extents. Z_2 and Z_3

c. Cuboid.

Fig 3.3 Discrete group of cuboids of the $E-D$ part
shown in Fig 3.2a

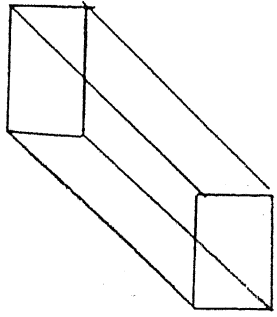


a. Orthographic View

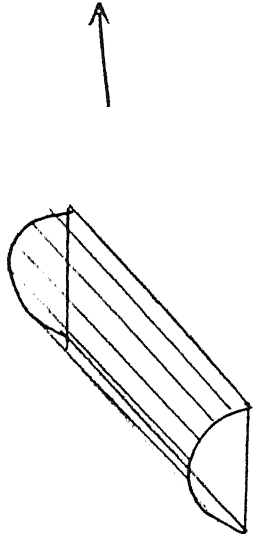


b. Elevation

Fig 3.4 A 3-D part with a hole feature running in two cuboids of different Z-extents



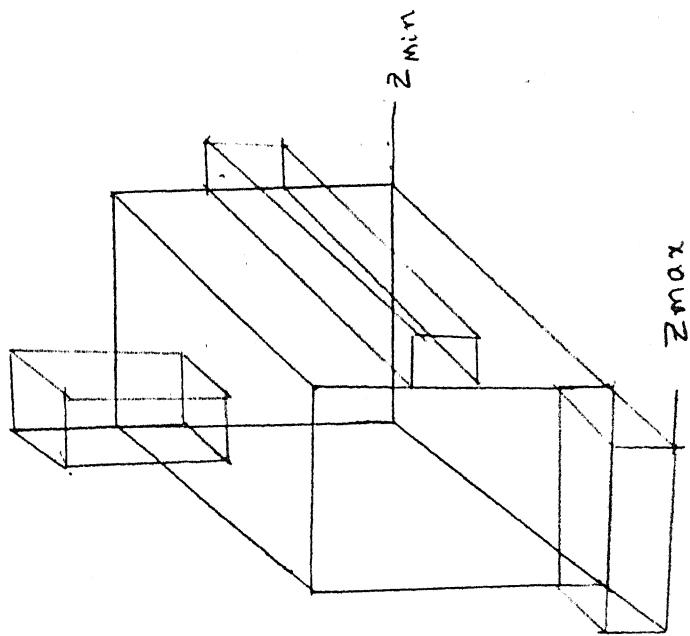
—



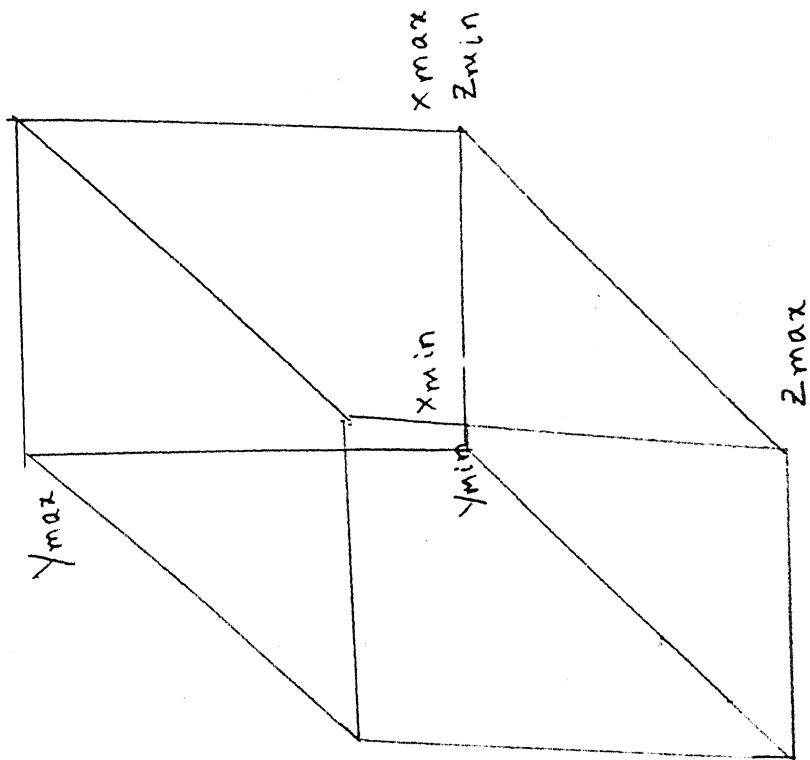
Partial cylinder

Enclosing cuboid

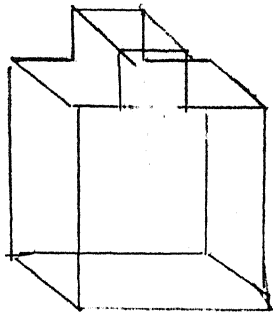
Fig 3.5 The set difference of an enclosing cuboid and a partial cylinder



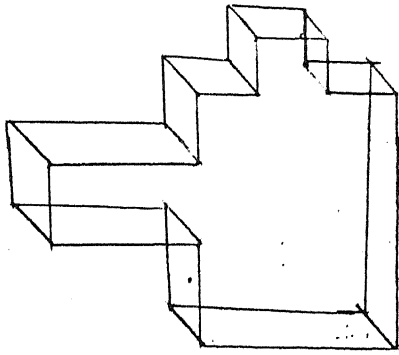
a. Resultant 3-D part
of Fig 3.2 a



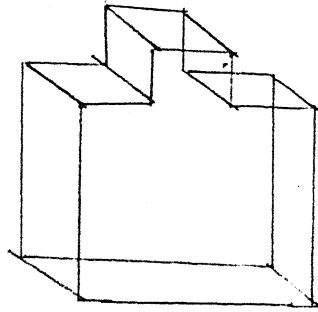
b. Stock cuboid



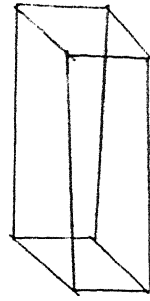
2.5-D part from Z_1 to Z_2



2.5-D part from Z_2 to Z_3



2.5-D part from Z_3 to Z_4



2.5-D part from Z_4

a. 2.5-D Part series

Fig 3.6 A resultant 3-D part, stock and 2.5-D part series of the resultant 3-D part

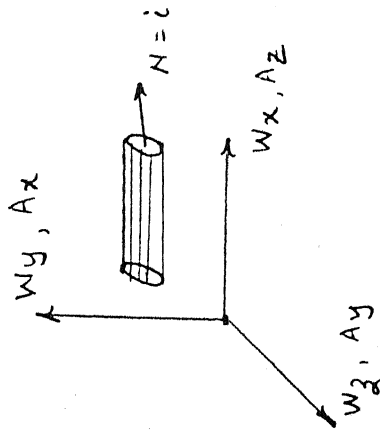


Fig 3.7 A cylinder with a positive X-axis extrusion

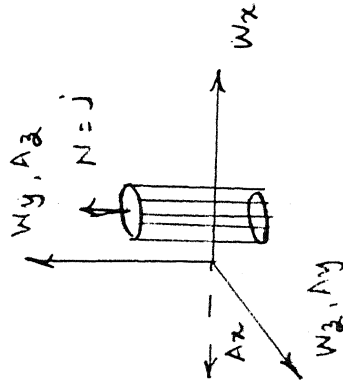


Fig. 3.9 A cylinder with a positive Y-axis extrusion

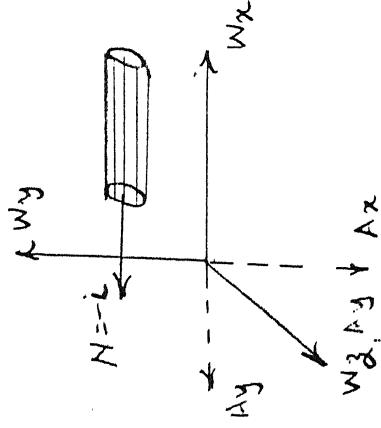


Fig 3.8 A cylinder with negative X-axis extrusion

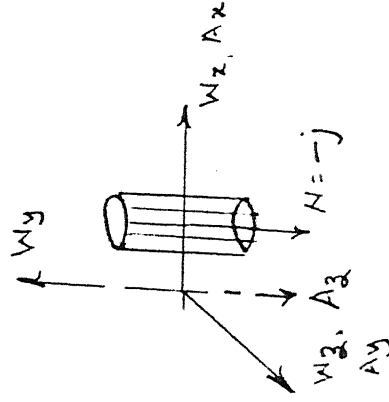
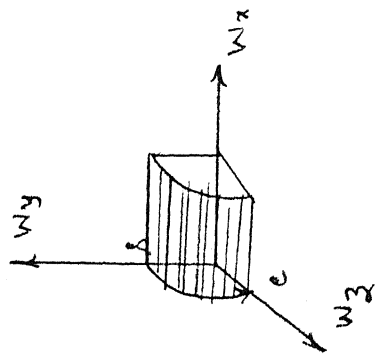


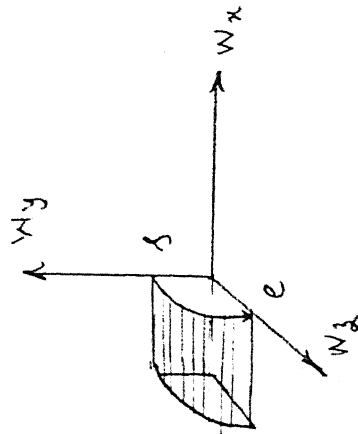
Fig 3.10 A cylinder with negative Y-axis extrusion



a. WCS

b. ECS

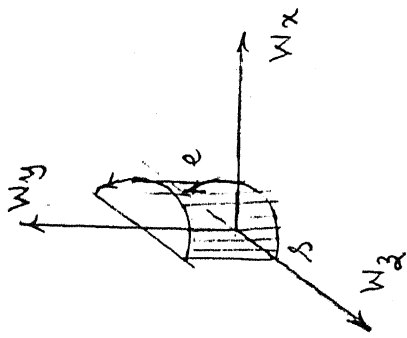
Fig 3.11 A partial cylinder with positive X-axis extrusion



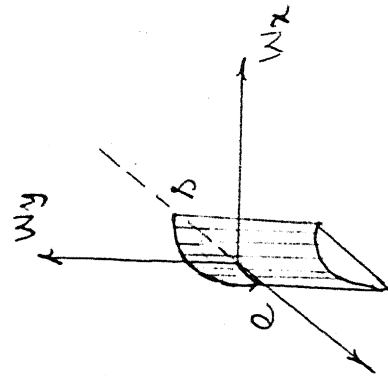
a. WCS

b. ECS

Fig. 3.12 A partial cylinder with negative X-axis extrusion



a. WCS

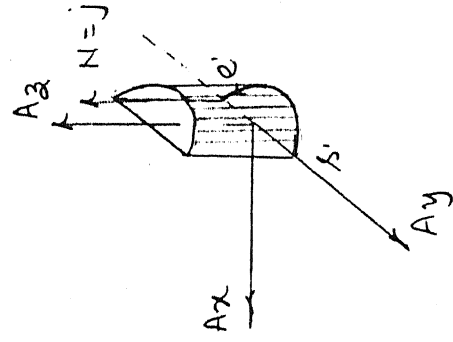


a. WCS

Fig 3.14

A partial cylinder with negative

y -axis



b. ECS

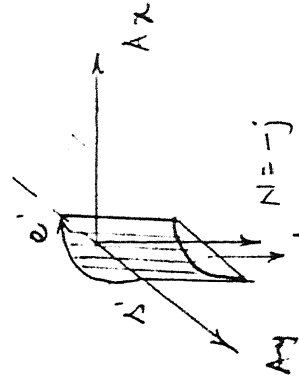
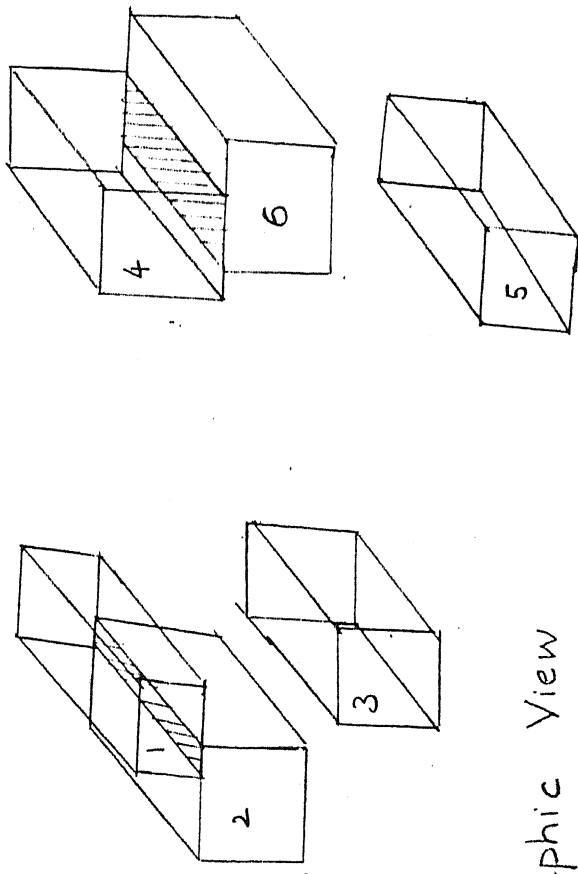
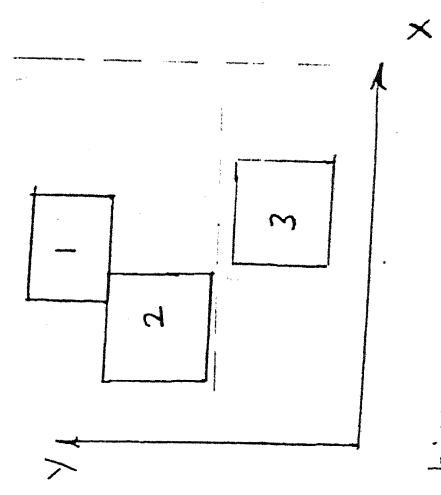


Fig 3.13 A partial cylinder with positive y -axis extrusion

with negative



a. Orthographic View



b. Elevation

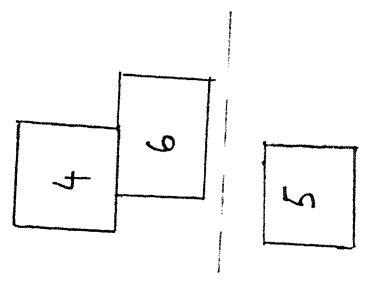


Fig 3.15 Schematic illustration of adjoining analysis

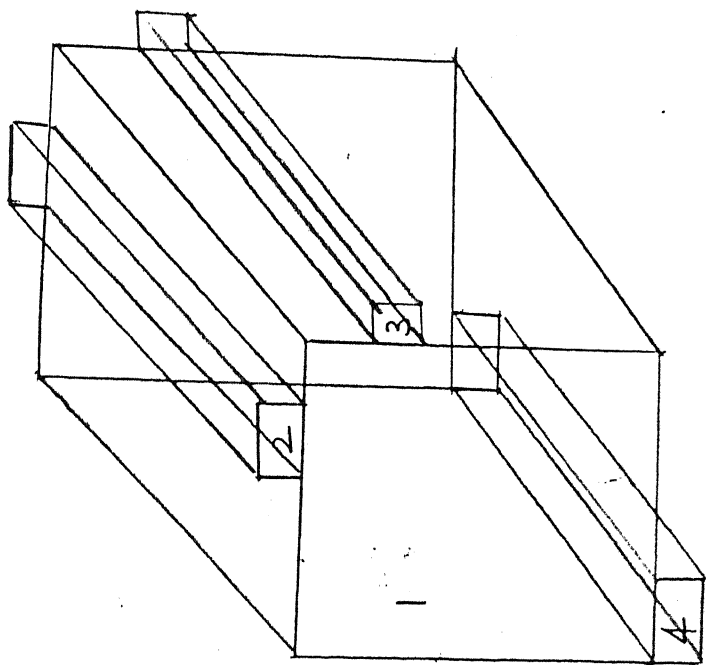
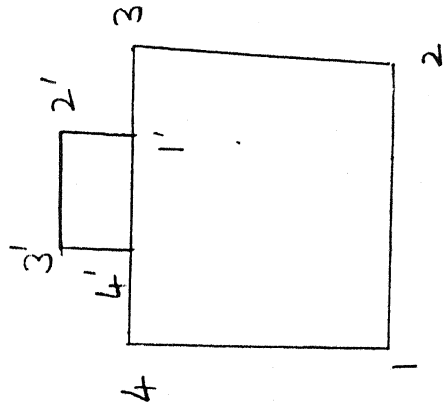
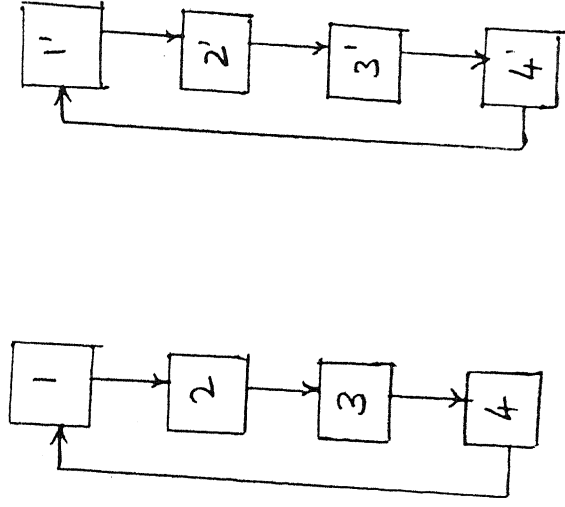


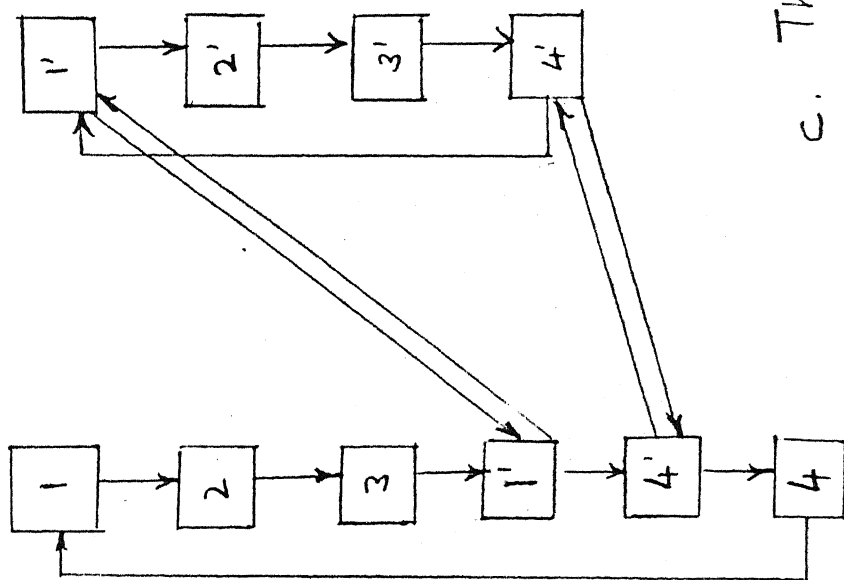
Fig 3.16 A group of mergeable cuboids with same z - extents



a. Base planes of the solids

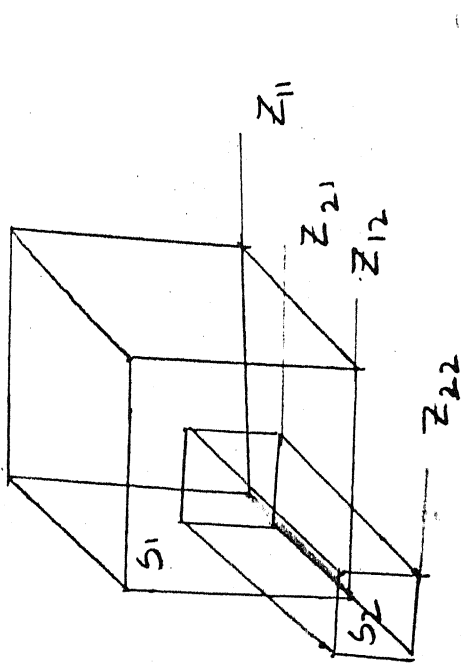


b. The counter clockwise lists
vertices of the base planes

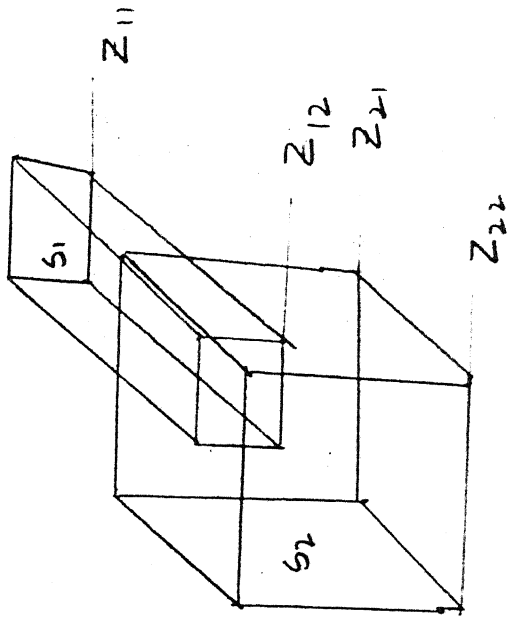
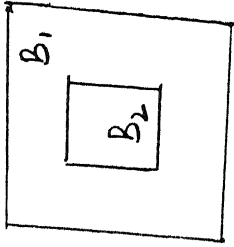


c. The vertex lists after adding the intersection points of the base plane:

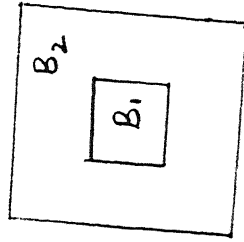
Fig 3.17 Merging of two cuboids of same Z - extents

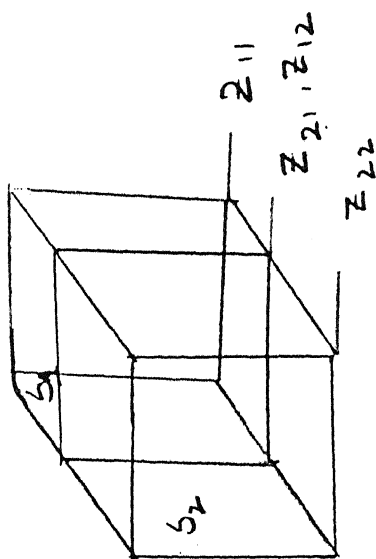


Case 1

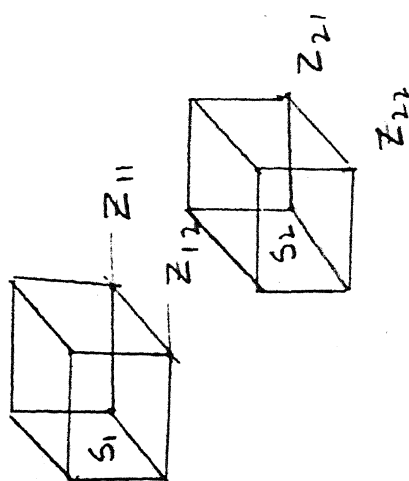
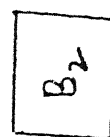
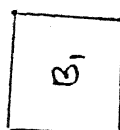
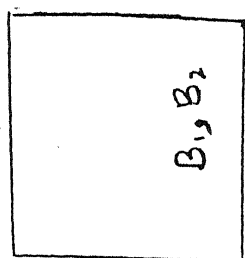


Case 2

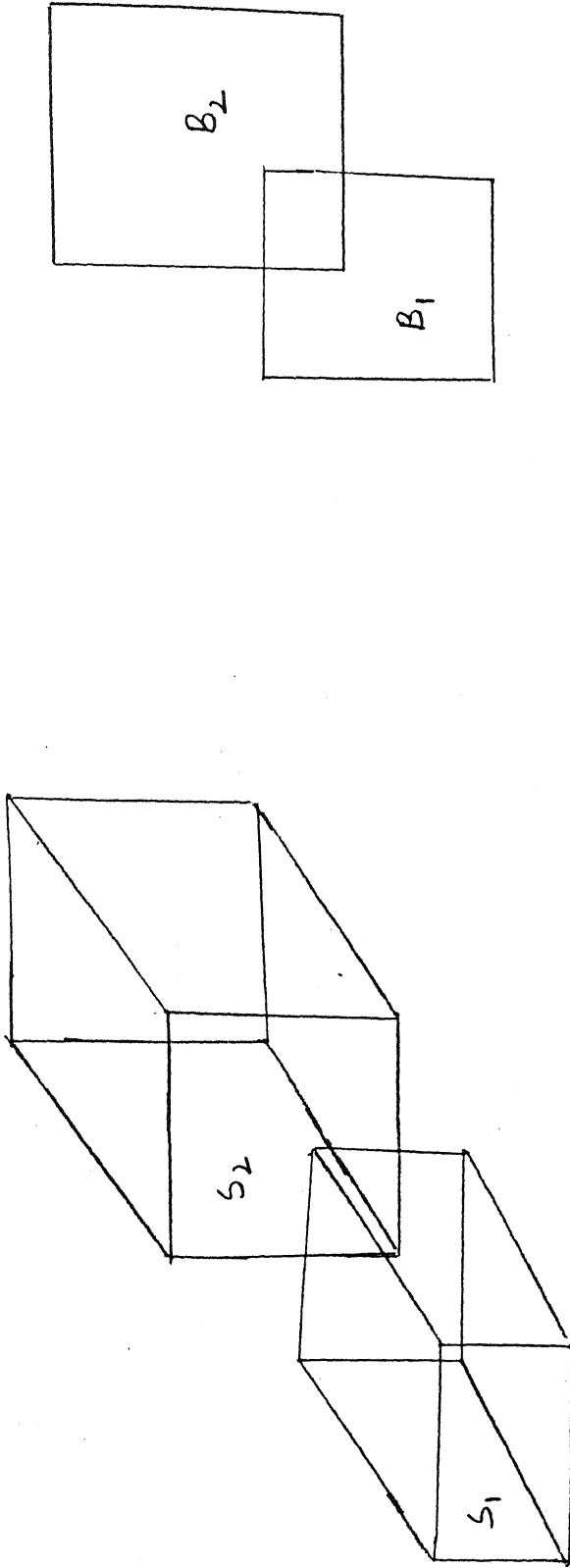




c. Case 3

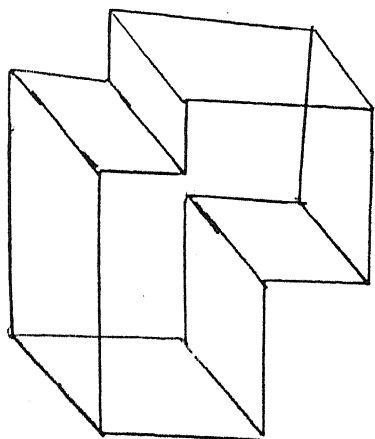


d. Case 4

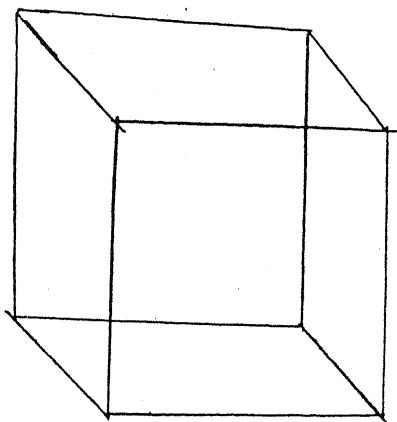


e. Case 5

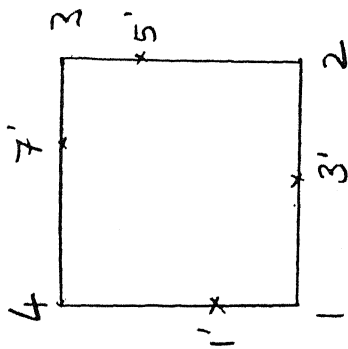
Fig 3.18 Different cases of merging of cuboids
along z -axis



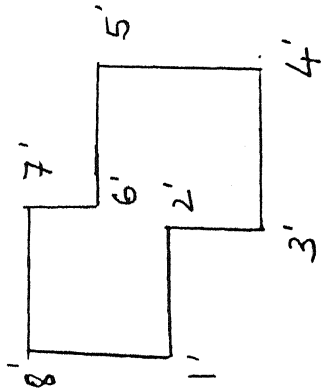
b. 2.5-D part



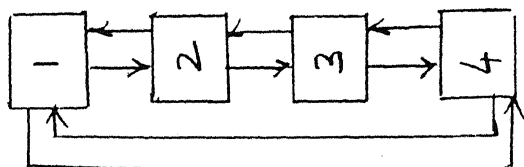
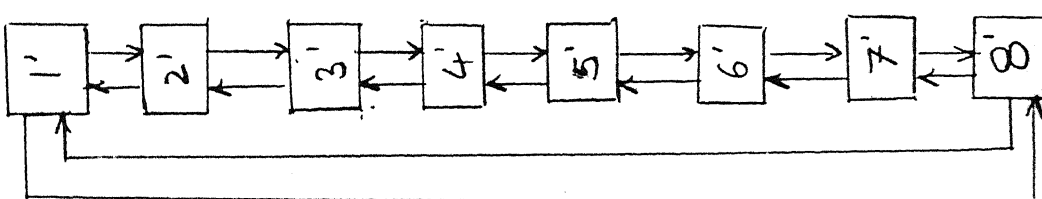
a. 2.5-D blank



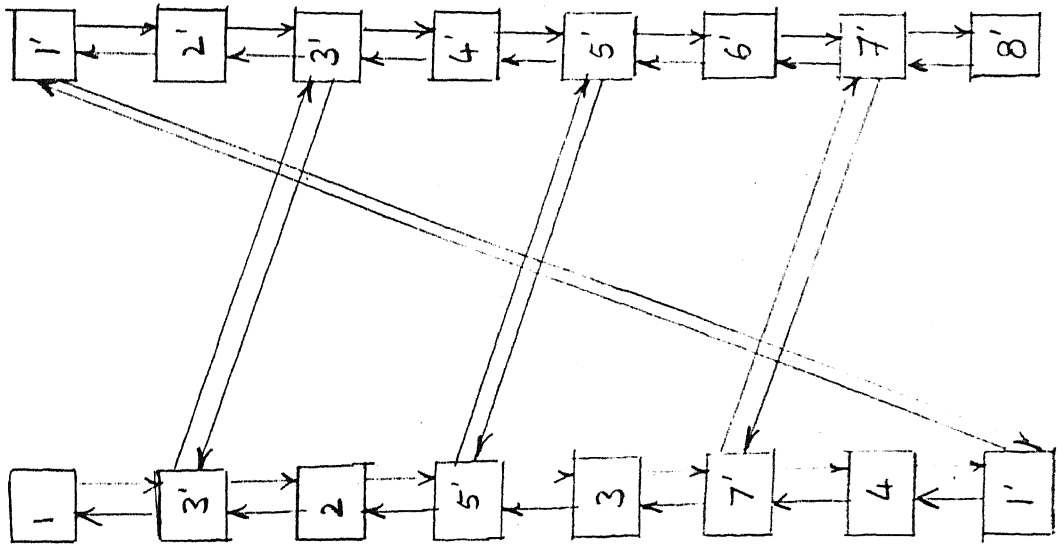
c. Base plane of the
2.5-D blank



d. Base plane of the
2.5-D part

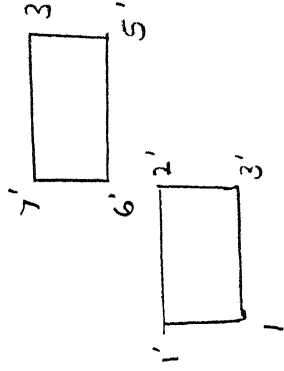


e. Vertex lists of the
base planes

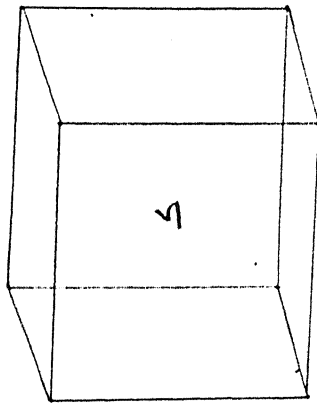


f. Vertex Lists after adding the intersection points

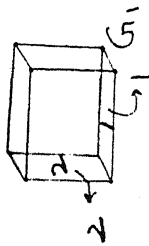
Fig 3.19 Schematic illustration of Weiler - Atherton find the set difference of 2.5-D



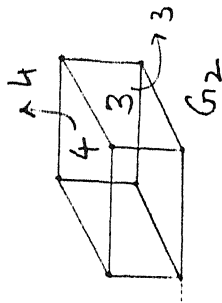
g. The set difference of the base planes

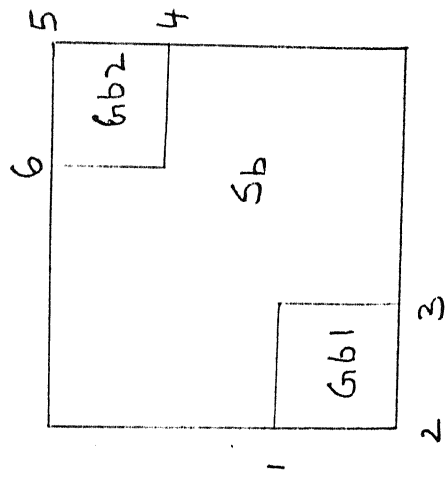


a. 2.5 - D stock



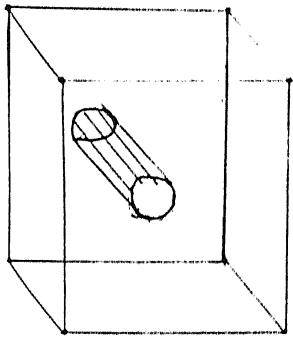
b. Ghost volumes



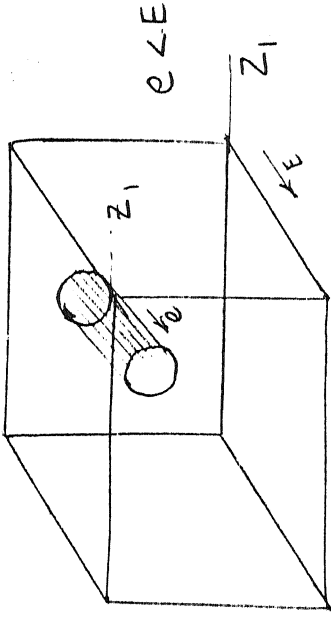


c. The base planes of the stock and ghost volumes

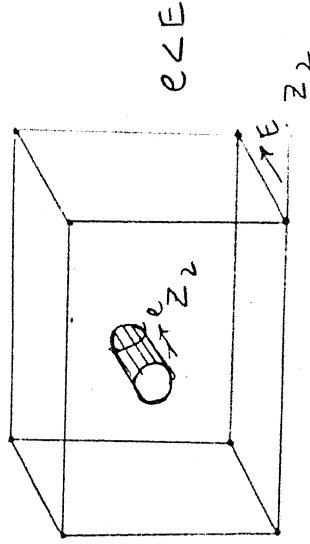
Fig 3.20 Tool accessibility of ghost volumes



a. A through hole



b. A blind hole opening
on XY -plane



c. A blind hole opening on
 XY -plane ; $z = z_2$

Fig 3.21 Top. accessibility of the holes

Chapter IV

SOFTWARE DEVELOPMENT AND EXAMPLES

The *feature recognition* algorithms discussed in Chapter II and Chapter III are implemented using Turbo C 2.0 in DOS 4.0 environment on a PC-AT. The implementation for 2.5-D parts and 3-D parts is carried out separately.

4.1 IMPLEMENTATION FOR 2.5-D PARTS

The system flow chart for 2.5-D parts is given Fig 4.1. The main modules in the implementation are

1. Input module
2. Feature recognition and machinability analysis module
3. Output module

4.1.1 Input Module

In this module, 2.5-D part is modeled interactively by the user as a polyline with certain thickness. A DXF file of the model is created by the name *autout.dxf*. Generally, a DXF file contains a lot of information besides the geometry of the model. But the DXF file for the present system should contain geometrical information only. AutoCAD facilitates to generate such a DXF file.

4.1.2 Feature Recognition and Machinability Analysis Module

In this module, the user generated *autout.dxf* file is read, and the *ghost volumes* present in the part are identified. Each

ghost volume is analyzed for features and features are checked for machinability.

The various functions in this module are explained below

Function *gdata_extraction* reads the DXF file and stores the input into a doubly linked list.

Function *classify_vertices* identifies the head and deflection points and determines whether the user entered the vertices of the polyline in clock-wise order or anti clock-wise order.

If the polyline is defined by the vertices in the anti clock-wise order, the Function *convert_the_list* sets the forward direction of the list to clock-wise direction by reversing the initial direction of the linked list. It divides the linked list into four parts one for each side i.e. left, top, right and bottom.

Function *find_ghost_volumes* identifies the *ghost volumes* on all four sides of the part by analyzing the linked lists of each side.

Function *feature_recognition1* extracts the complete features, t-slot, dovetail and v-slot present in each *ghost volume*. The different features are stored in different lists.

Function *feature_recognition2* extracts the partial features, partial dovetails and partial t-slots from each *ghost volume*. These features are stored in separate lists.

Function *find_vertslots* divides the *ghost volumes* into vertical slots. Function *find_horislots* divides the *ghost volumes* into horizontal slots.

Function *merge_partial_features* merges the matching partial

features into single complete feature. The matching partial features are deleted from their lists and the unified complete feature is added to the corresponding list.

Function *machinability_1* checks the machinability of complete features, t-slots, dovetails, reverse dovetails and v-slots.

Function *machinability_2* checks the machinability of partial features, partial dovetails and partial reverse dovetails.

Function *machinability_3* checks the machinability of partial features, partial t-slots and horizontal partial reverse dovetails.

The aggregate code of the functions is too large to be written in a single file, since TurboC has a limit on the maximum size of a file. Hence modular approach is followed. In modular approach the different functions of a program are written in different files and all the files are combined under the name of a project file (similar to a make file on Unix C). The different files in the present system are explained below.

File1.c contains the functions *gdata_extraction*, *classify_the_vertices*, *convert_the_list*, *find_ghost_volumes* and *feature_recognition_1*.

File2.c contains the function *feature_recognition_2*

File3.c contains the functions *find_vertslots* and *find_horislots*.

File4.c contains the function *merge_partial_features*.

File5.c contains the function *machinability_1*.

File6.c contains the function *machinability_2*.

File7.c contains the function *machinability_3*.

Fig 4.2 shows the program control flow and task flow of the *feature recognition* and *machinability analysis* module.

As shown in Fig 4.1 the program uses cutting tool database in checking the machinability of the *features*. The database stores the geometrical specifications for standard tools of different kinds.

4.1.2.1 Database for T-slot Milling Cutters

The database for t-slot cutters is shown in the form of Table 4.1. The terminology, used in the table, refers to Fig 2.14b in chapter-II.

4.1.2.2 Database for Dovetail Milling Cutters

The database for dovetail cutters is shown in the form of Table 4.2. The terminology, used in the table, refers to Fig 2.16b in chapter-II.

4.1.2.3 Database for Reverse Dovetail Milling Cutters

The database for reverse dovetail cutters is shown in the form of Table 4.3. The terminology, used in the table, refers to Fig 2.18b in chapter-II.

4.1.2.4 Database for Single Angle Milling Cutters

The database for single angle milling cutters is given in Table 4.4. The terminology, used in the table, refers to Fig 2.23b in chapter-II.

4.1.2.5. Database for equal angle milling cutters

The database for equal angle milling cutters, v-slot cutters, is given in Table 4.5. The terminology, used in the table, refers to the Fig 2.19b in Chapter-II.

All linear measures are in mm and angular measures are in degrees.

4.1.3 Output Module

In this module features on all four sides of the part are displayed, in an arbitrary order, along with the information about their machinability. If a feature is non-machinable, then the reason for its non-machinability is displayed, i.e. whether the cutting tool is not available or the tool interferes with the part. If a feature is machinable, then the cutting tool number and the type of the spindle to be used, i.e. horizontal or vertical, are displayed.

The module is coded using Turbo Graphics. Different features are displayed in different colors.

4.2 IMPLEMENTATION FOR 3-D PARTS

The system flow chart for 3-D parts is shown in Fig 4.3. The main modules in the implementation are :

1. Input module
2. Feature recognition module
3. Output module

4.2.1 Input Module

The input module is similar to the one in the implementation of 2.5-D parts. The user models the 3-D parts as an ensemble of

primitives and creates a DXF file, which contains geometrical information only.

4.2.2 Feature Recognition Module

This module identifies the *cavity volume*, to be removed from the block, in the form of local and global *ghost volumes*. The various functions in this module are explained below.

Function *read_the_data* reads the geometry of the primitives of which the 3-D part is made up of and transforms the geometry of all primitives to WCS coordinate system of AutoCAD.

Function *enclose_the_primitives* encloses the non-hole primitives i.e. non-cylindrical primitives other than cuboids with minimum enclosing cuboids. The set difference of a primitive and its enclosing cuboid is a local *ghost volume*.

Function *find_Z_holes* merges the cuboids and classifies the holes which are parallel to Z-axis as through or blind holes. In case of blind holes the opening face of the hole, the tool accessible face, is found out.

Function *find_Z_slots* identifies the global *ghost volumes* in the part along Z-axis, i.e. the division of 3-D part into 2.5_D chips is carried out along Z-axis.

Function *find_Y_holes* classifies the holes, parallel to Y-axis as blind or through and finds out the tool accessible faces.

Function *find_X_holes* classifies the holes, parallel to X-axis as blind or through and finds out the tool accessible faces.

Fig 4.4 shows the program structure and control flow of the *feature recognition* module.

4.2.3 Output Module

The output of the system is partly alpha-numeric and partly graphic. The information about the local ghost volumes is given in the form of alpha-numeric output and global ghost volumes are displayed using Turbo Graphics.

4.3 AN EXAMPLE OF A 2.5-D PART

This section demonstrates an illustration for feature recognition of a 2.5-D part in the proposed system.

Fig 4.5 shows a 2.5-D part, modeled in AutoCAD. The part modeling and invoking the feature recognition program is explained in Section 4.4.

Fig 4.6 shows the base plane of the 2.5-D part with the t-slots on left side. The information about the machinability of each t-slot is also given. The size of the stock is also displayed.

Fig 4.7 shows the base plane of the 2.5-D part with the complete features on all four sides.

Fig 4.8 shows the base plane of 2.5-D part with complete and partial features recognized. Different features are displayed in different colors.

Fig 4.9 shows the base plane with all features including the slots.

4.4 USER'S MANUAL FOR 2.5-D FEATURE RECOGNITION SYSTEM

The following files should reside in the working directory for the successful functioning of the system :

file1.c	file2.c	file3.c	file4.c
---------	---------	---------	---------

```

file5.c      file6.c      file7.c      file8.c
file9.c      file10.c     variable.h   tstool.dat
dttool.dat   rdttool.dat   vstool.dat   hrdttool.dat
fearec2d.prj egavga.bgi    acad2d.bat

```

acad2d.bat should read as follows:

```
ECHO OFF
```

```
D:{the drive in which AutoCAD10 resides}
```

```
ACAD
```

```
E:{the drive of the working directory}
```

```
COPY D:\APPSOFT\ACAD10\AUTOUT.DXF {AutoCAD directory} E:
```

```
ECHO ON
```

4.4.1 Part Modeling

AutoCAD is invoked by typing acad2d at the DOS prompt.

```
C:\>ACAD2D <ENTER>
```

AutoCAD displays a menu, which looks as follows :

0. Exit AutoCAD
1. Begin a NEW drawing
2. EDit an EXISTING drawing
3. Plot a drawing
4. Printer plot a drawing
5. Configure AutoCAD
6. File utilities
7. Compile shape/font description file
8. Convert old drawing

To create a new drawing option 1 is selected and to retrieve an existing drawing option 2 is selected. Selection of either option makes the drawing editor of AutoCAD to appear along with

the customary screen and the prompt 'Command'.

The part is modeled by specifying the thickness of the polyline and then drawing the polyline. The dialogue with AutoCAD while modeling a part is explained below. The user responses are given in *italic capitals*.

Command : *SETVAR* <ENTER>

Variable name or ? : *THICKNESS* <ENTER>

New value for thickness <0.00> : *X.XX* <ENTER>

Command :

Command : *PLINE* <ENTER>

Then a polyline is defined as a closed sequence of lines. The holes and pockets are defined inside the boundary of polyline by drawing the entities circles and solids respectively. The thickness of the circles and solids is set by *SETVAR* command, before drawing them.

Command : *CIRCLE* <ENTER>

3p/2p/ttp/<center point> : *X,Y* <ENTER>

Diameter/<Radius> : *X.XX* <ENTER>

Command :

Command : *SOLID*

First point : *X1, Y1* <ENTER>

Second point : *X2, Y2* <ENTER>

Third point : *X3, Y3* <ENTER>

Fourth point : *X4, Y4* <ENTER>

Third point : <ENTER>

Command :

The above dialogues describe one way of drawing the entities, AutoCAD user's guide can be referred for alternate ways

of drawing them. After modeling the part the DXf file of the drawing is created by the following dialogue.

Command : DXFOUT <ENTER>

File name <drawing name> : AUTOUT <ENTER>

Enter decimal places of accuracy (0 to 16) / Entities /

Binary <6> : E <ENTER>

Select objects : C

First corner : 0, 0 <ENTER>

Second corner : 20, 20 <ENTER>

Enter decimal places of accuracy (0 to 16) / binary <0> :
X <ENTER>

Command :

The following command takes the control to the main menu of AutoCAD.

Command : END <ENTER>

Selecting the option 0 takes to the DOS prompt.

C:>\

If fearec3d.exe resides in the directory, then it is invoked by typing

C:>\FEAREC2D <ENTER> ,

else fearec.prj is compiled to fearec2d.exe before invoking it. The compilation is done as follows :

C:>\TC FEAREC2D.PRJ <ENTER>

The above command takes the control to Turbo C editor, with fearec2d.prj loaded in it. Pressing F9 compiles all the constituent files of fearec2d.prj and links them to make fearec2d.exe.

4.5 AN EXAMPLE OF A 3-D PART

In this section an example of *feature recognition* for a 3-D part is demonstrated.

Fig 4.10 shows a 3-D part modeled in AutoCAD. The 3-D part is modeled as an ensemble of primitives, drawing entities in AutoCAD. The modeling of the part and invoking the *feature recognition* is explained in Section 4.6.

Fig 4.11 shows the alpha-numerical output about the blind/through holes in the 3-D part. The information about the tool accessibility the holes is also displayed.

Fig 4.12 shows the graphical output of the global *ghost volumes*. The *ghost volumes* are shown in alternate colors for clarity of display.

4.6 USER'S MANUAL FOR 3-D FEATURE RECOGNITION SYSTEM

The following files should reside in the working directory for the successful functioning of the system :

```
file21.c      file22.c      file23.c      file24.c
file25.c      file26.c      var3d.h
fearec3d.prj  egavga.bgi
acad3d.bat
```

acad3d.bat should read as follows :

```
ECHO OFF
```

```
D:{the drive in which AutoCAD10 resides}
```

```
ACAD
```

```
E:{the drive of the working directory}
```

```
COPY D:\APPSOFT\ACAD10\ACAD3D.DXF {AutoCAD directory} E:
```

4.6.1 Part Modeling

AutoCAD is invoked by typing acad3d at the DOS prompt.

C:\>ACAD3D <ENTER>

AutoCAD displays a menu, which is described in Section 4.4.1. Since a 3-D part is modeled as an ensemble of drawing entities solids, arcs and circles, the dialogues with the AutoCAD to draw a solid and a circle are explained in Section 4.4.1. The dialogue to draw an arc is explained below.

Command : ARC <ENTER>

Center/<Start point : X_1 , Y_1 <ENTER>

Center/End/<Second point : C <ENTER>

Center : X_2 , Y_2 <ENTER>

Angle/Length of chord/<end point> : A <ENTER>

Included angle : A_1 <ENTER>

Command :

The thickness of a drawing entity can be set by the command SETVAR which is explained in Section 4.4.1 before drawing it or it can be varied by the command CHPROP after drawing it. The dialogue for CHPROP is given below.

Command : CHPROP <ENTER>

Select objects : {the objects can be selected by clicking the mouse on them or giving a window/cross size such that the objects lie in side the window/cross (refer AutoCAD user's manual for more details) }

Select objects : <ENTER>

Change what property (Color/Layer/LType/Thickness)? : T
<ENTER>

New thickness <default thickness> : X <ENTER>

If *fearec3d.exe* resides in the directory, then it is invoked by typing

```
C:>\FEAREC3D <ENTER>,
```

else it is created similar to *fearec2d.exe* before invoking it.

4.7 Additional Examples

Fig 4.13 shows an AutoCAD model for a 2.5-D part, which is more complicated than the part shown in Fig 4.5. Fig 4.14 shows various features recognized in the 2.5-D part shown in Fig 4.13.

Fig 4.15 is an AutoCAD model of one of the 3-D parts, for which the implementation has been tested. It is more complicated than the part in Fig 4.9.

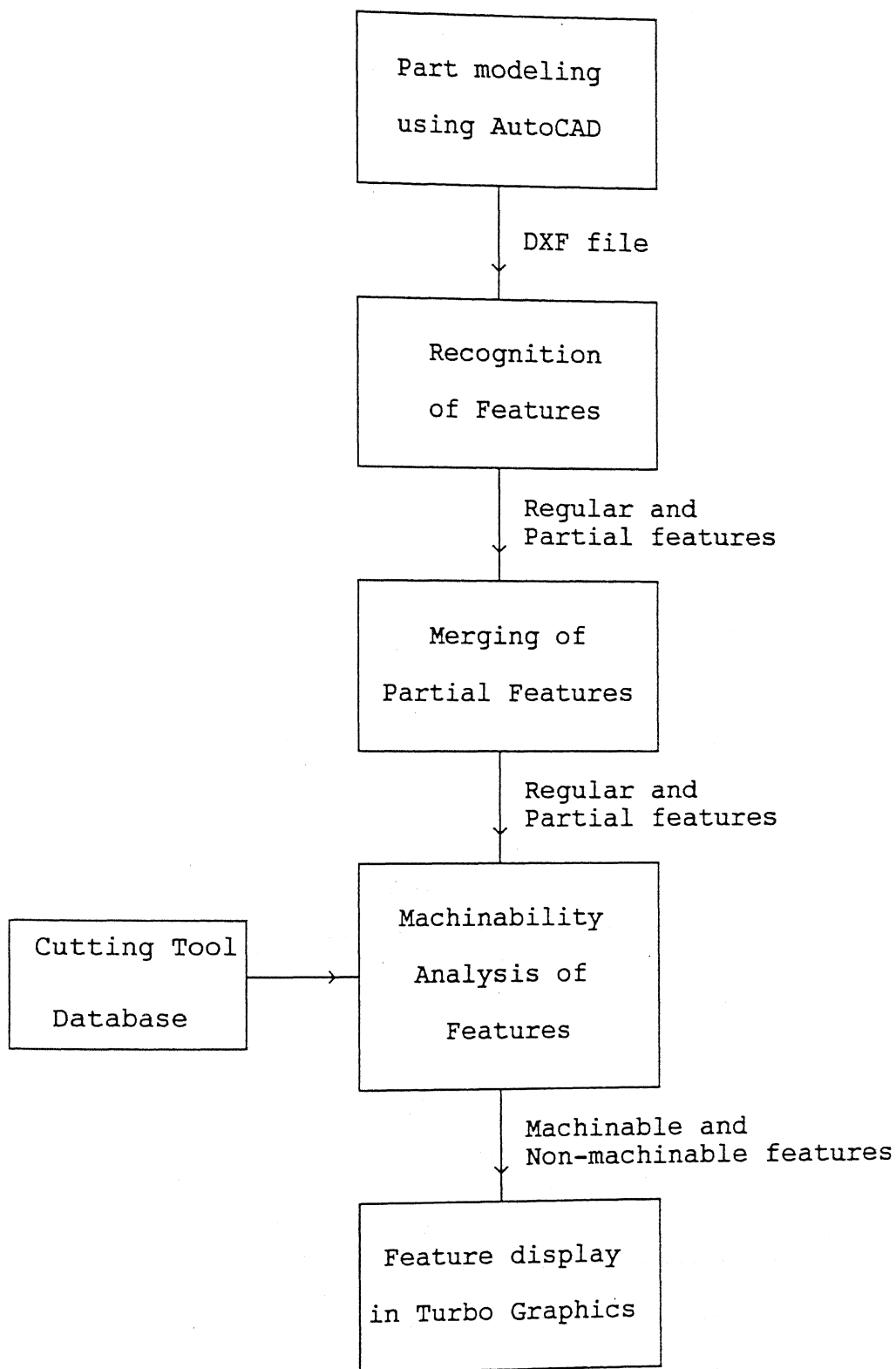


Fig 4.1. System flow chart for feature recognition of 2.5-D part

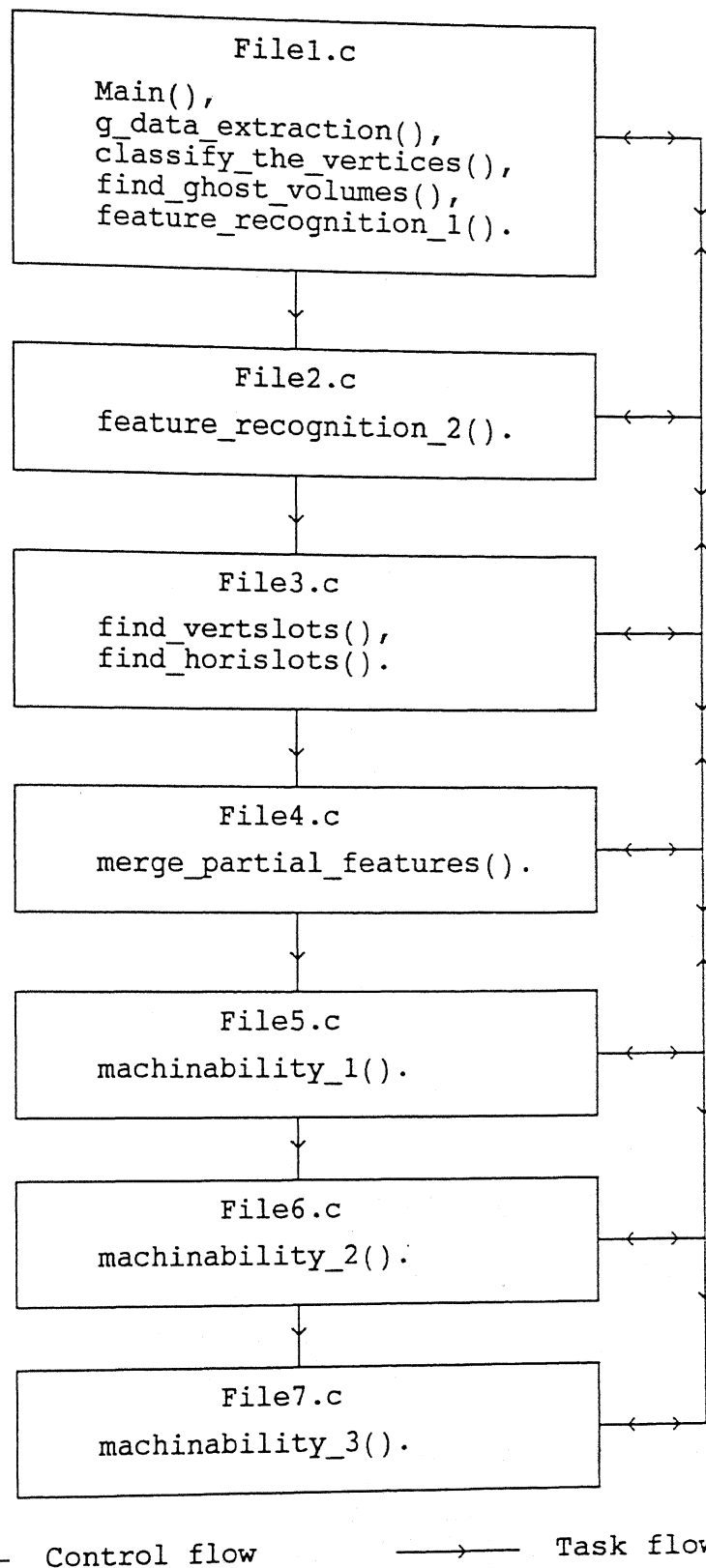


Fig 4.2. The program flow chart for feature recognition and machinability analysis module

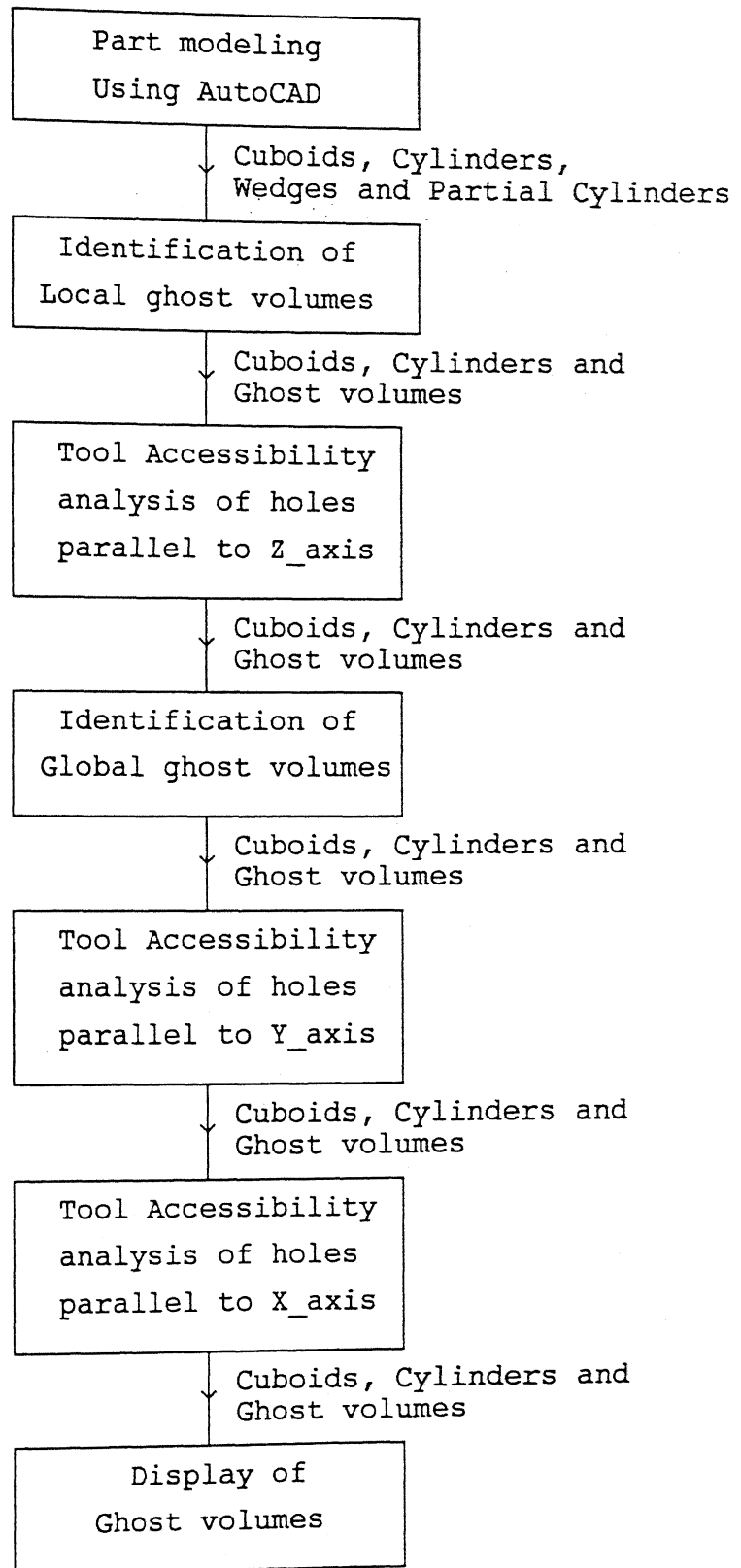


Fig 4.3 System flow chart for feature recognition of 3-D parts

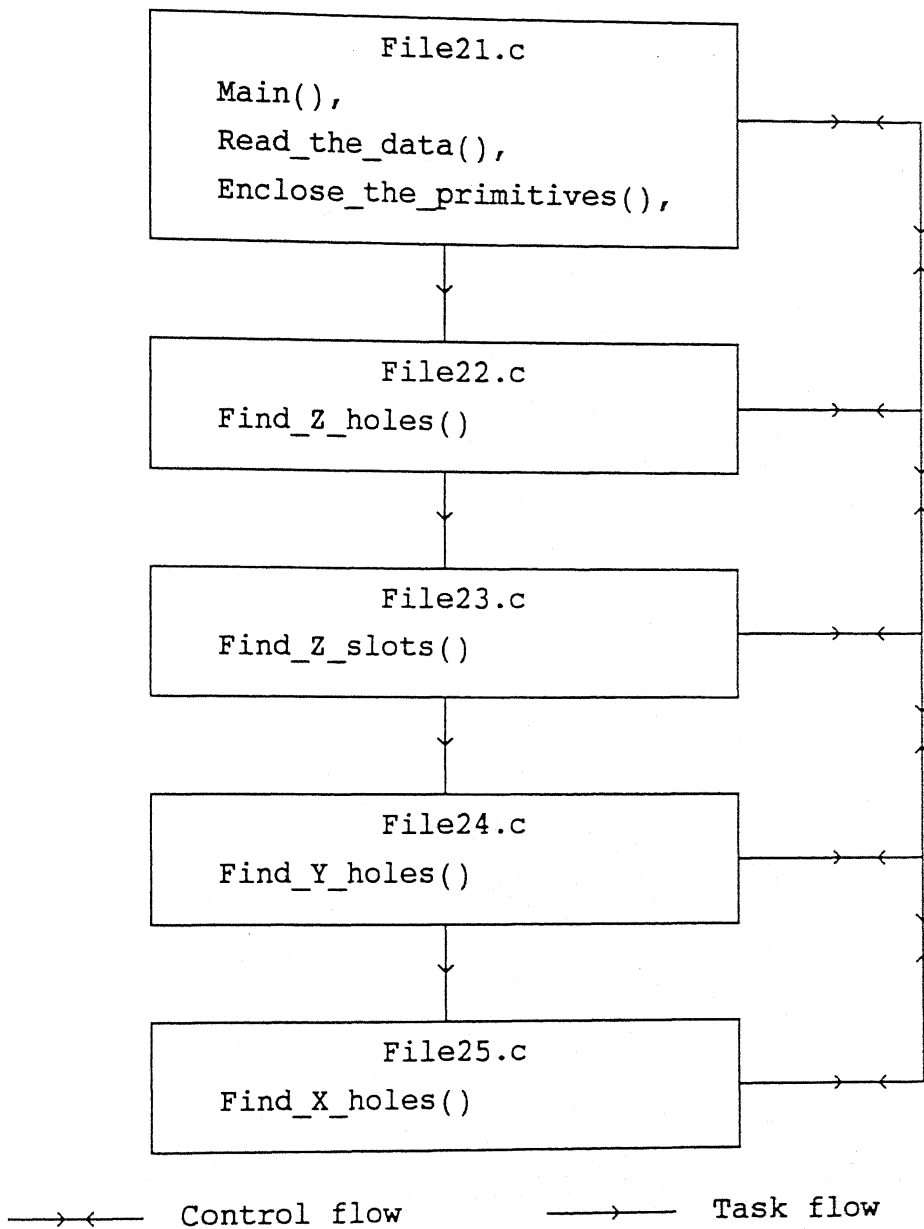


Fig 4.4 Program flow chart for feature recognition module

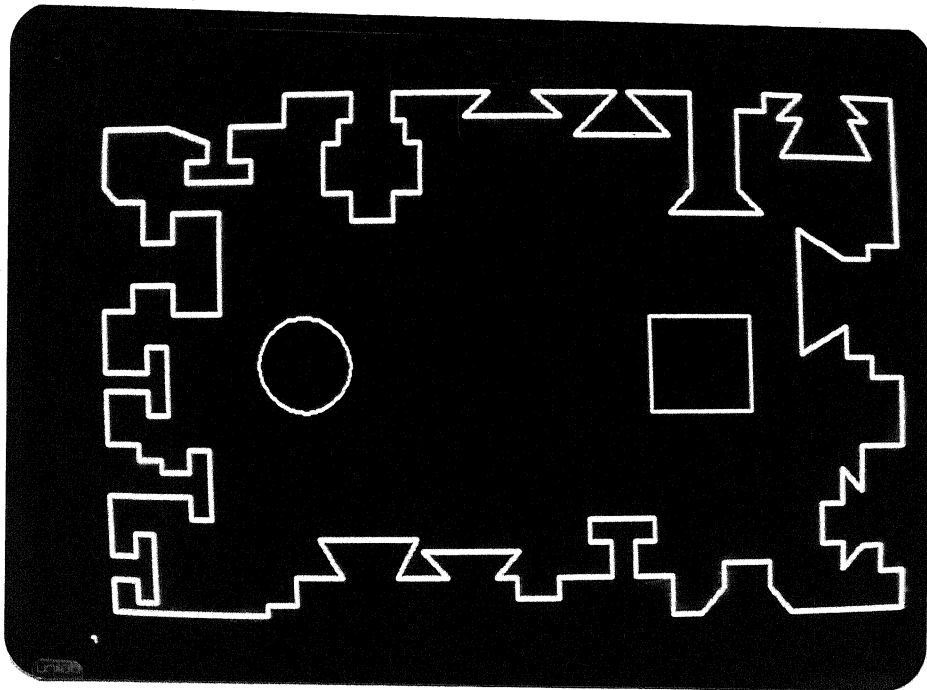


Fig 4.5 AutoCAD model of a 2.5-D part

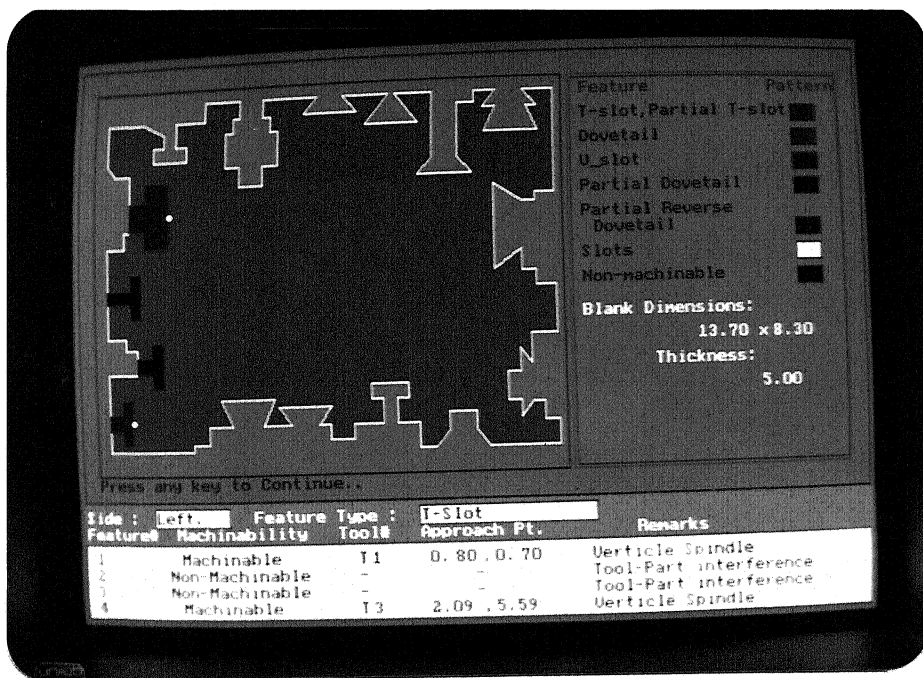


Fig 4.6 Base planes corresponding to Fig 4.5 with
T-slots on left side

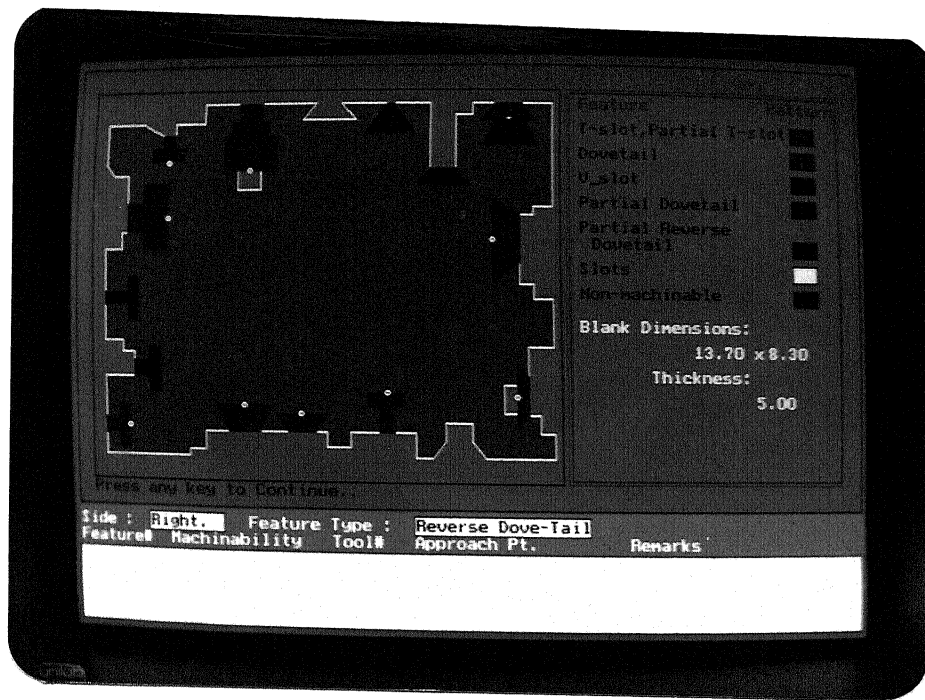


Fig 4.7 Base plane corresponding to Fig 4.5 with complete features

Fig 4.8 Base plane corresponding to Fig 4.5 with complete and partial features

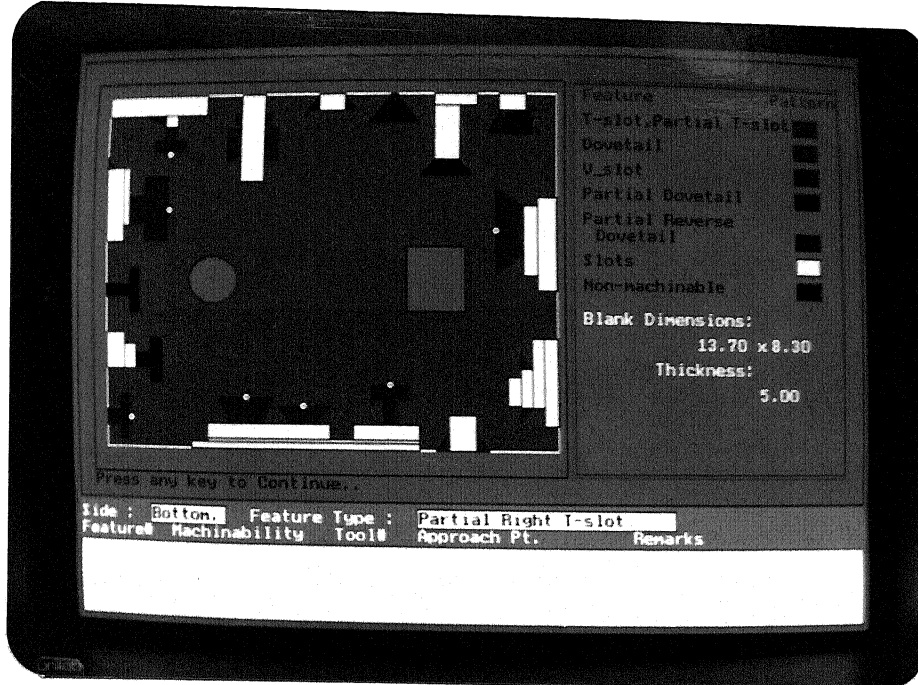


Fig 4.9 Base plane corresponding to Fig 4.5 with all features including slots

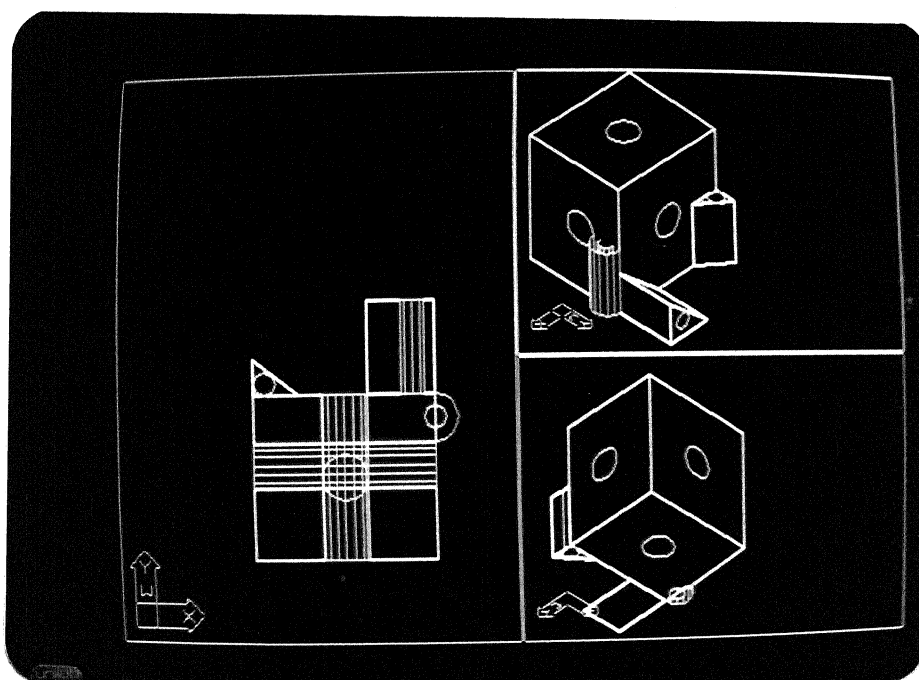


Fig 4.10 AutoCAD model of 3-D part shown in three views

Holes found in the part.										
S.No.	X center	Y center	Z center	Radius	Extrusion Vector			Blind/Through	Direction	
					Length	i	j	k		
1	3.50	5.50	0.0	0.25	2.00	0.0	0.0	1.00	T	1
2	5.25	3.50	0.0	0.50	4.00	0.0	0.0	1.00	T	1
3	7.25	4.75	0.0	0.25	2.00	0.0	0.0	1.00	B	1
4	5.25	1.75	2.00	0.50	3.50	0.0	1.00	0.0	T	1
5	6.75	7.25	0.50	0.25	2.00	0.0	1.00	0.0	B	-1
6	3.25	3.75	2.00	0.50	4.00	1.00	0.0	0.0	T	1

Fig 4.11 Alpha-numeric output of the hole information in 3-D part shown in Fig 4.10

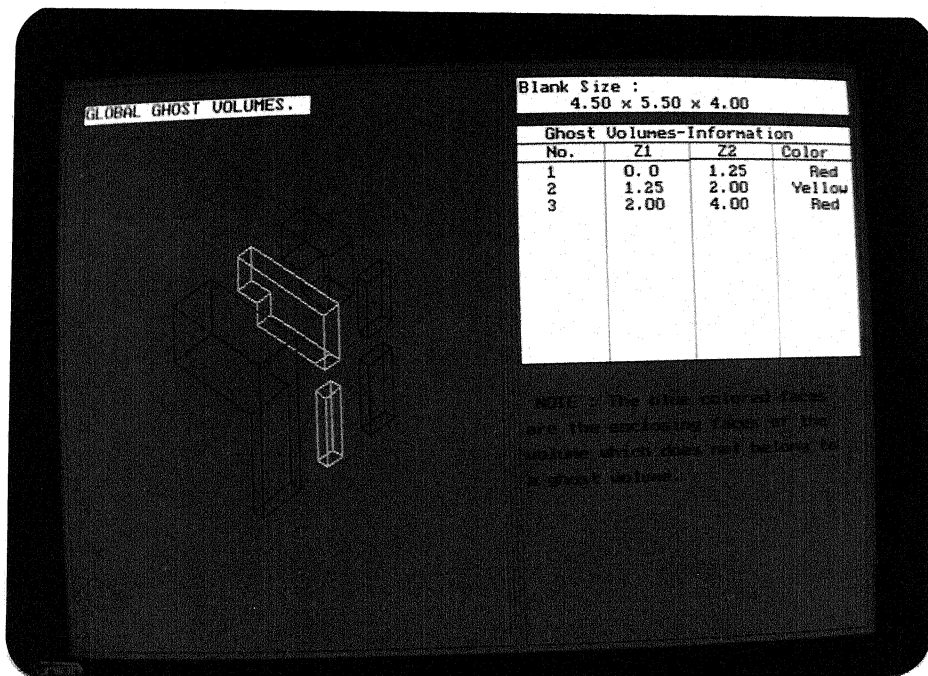


Fig 4.12 Graphical output of the global ghost volumes in 3-D part shown in Fig 4.10

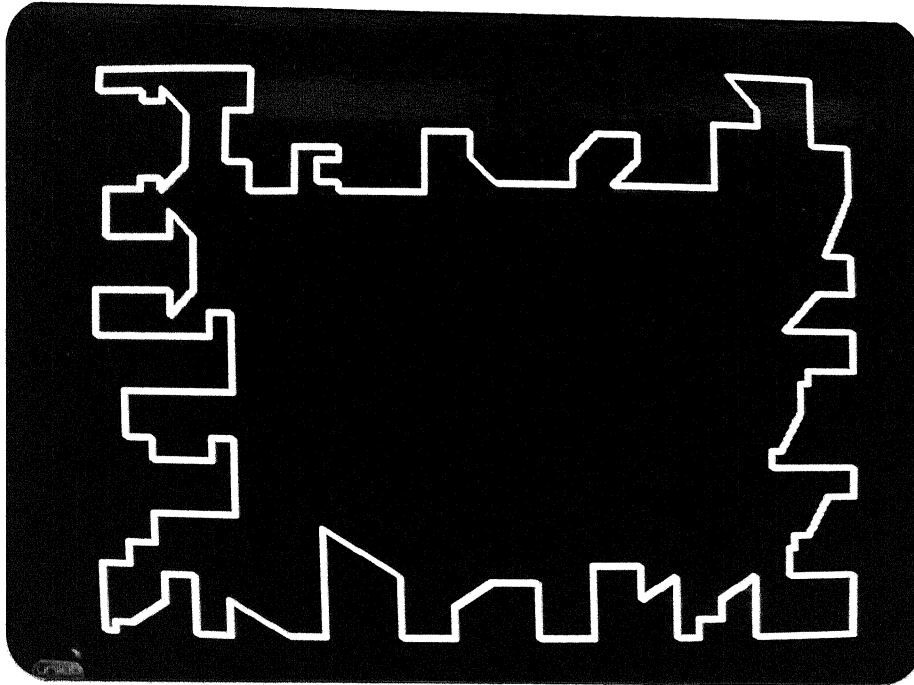


Fig 4.13 AutoCAD model of a complicated 2.5-D part

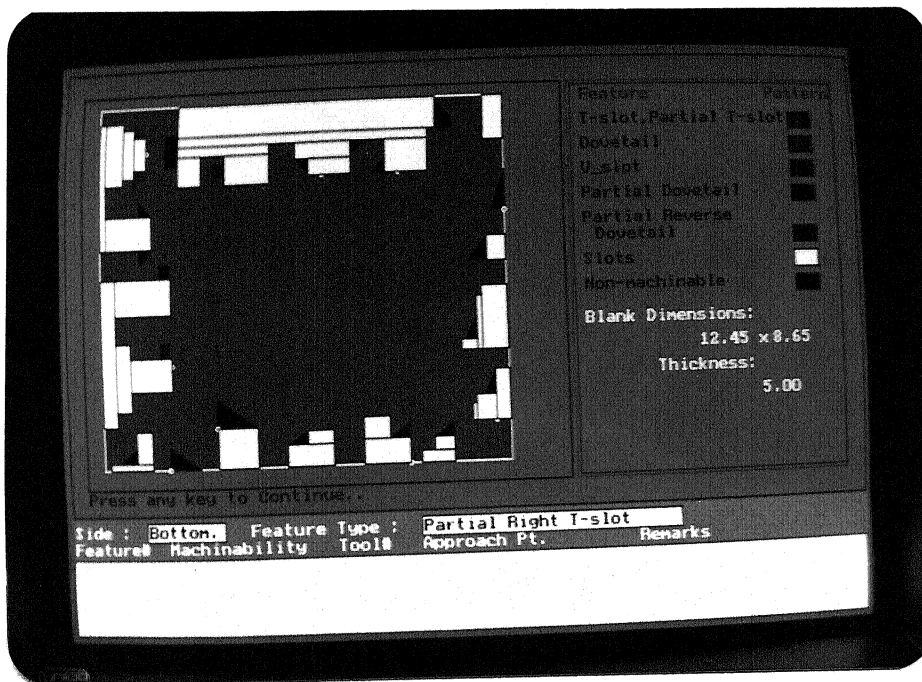


Fig 4.14 Various features in the 2.5-D part shown in Fig 4.13

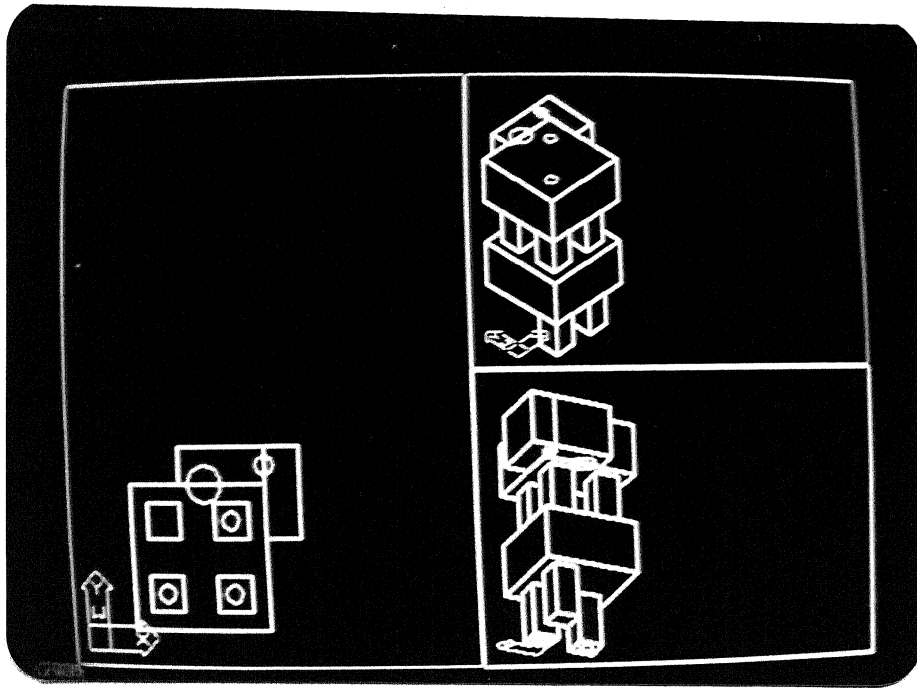


Fig 4.15 AutoCAD model of a complicated 3-D part

Table 4.1 Specifications for t-slot milling cutter.

tb_d	tb_t	tn_d	tn_l	ts_d	tt_l
11	3.5	4	10	10	53.5
12.5	6	5	11	10	57
16	8	7	14	10	62
18	8	8	17	12	70
21	9	10	20	12	74
25	11	12	23	16	82
32	14	15	28	16	90
40	18	18	34	25	108
50	22	25	42	32	124

Table 4.2 Specifications for dovetail milling cutter.

α_t	tb_d	tne_d	ts_d	ts_l	tt_l	tb_t
45	16	6	12.5	40	60	4
45	20	7	12.5	40	70	5
45	25	8	12.5	40	80	6
60	16	6	12.5	40	60	6
60	20	8	12.5	40	70	8
60	25	9	12.5	40	80	10
70	16	8	12.5	40	60	7
70	20	9	12.5	40	70	9
70	25	9	12.5	40	80	11

Table 4.3 Specifications for reverse dovetail milling cutter.

α_t	tt_d	tb_d	tne_d	ts_d	ts_l	tt_l	tb_t
45	16	8	6	12.5	40	60	4
45	20	10	7	12.5	40	70	5
45	25	13	8	12.5	40	80	6
60	16	9	6	12.5	40	60	6
60	20	11	8	12.5	40	70	8
60	25	14	9	12.5	40	80	10
70	16	11	8	12.5	40	60	7
70	20	13	9	12.5	40	70	9
70	25	17	9	12.5	40	80	11

Table 4.4 Specifications for single angle milling cutter.

te_d	ti_d	tb_w	α_t
50	16	12	60, 65, 70, 75, 80, 85.
63	22	18	60, 65, 70, 75, 80, 85.
63	22	26	70, 75, 80.
80	22	32	70, 75, 80.
100	27	36	70, 75, 80.

Table 4.5 Specifications for equal angle milling cutter.

te_d	α_t	tb_w	ti_d
56	45	10	16
56	60	12	16
56	90	14	16
65	45	12	22
65	60	16	22
65	90	18	22
80	45	16	22
80	60	20	22
80	90	22	22
100	45	18	27
100	60	25	27
100	90	28	27

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

In the present work, feature recognition algorithms have been developed and implemented for 2.5-D and 3-D prismatic parts. The algorithms are based on the concept that while extracting the machinable features from a part the *cavity volume* to be removed from the stock is required to be analyzed rather than the volume of the part. The *cavity volume* is conceived as the sum total of discrete volumes called *ghost volumes* and the approach is called the *ghost volume approach*.

The purpose of the present work is to study the feasibility of *ghost volume approach* in the feature recognition and subsequently in process planning of 2.5-D and 3-D prismatic parts. It is found that the approach is a promising one and can be used to develop generic algorithms for the feature recognition. Though the approach does not simplify the feature recognition of axisymmetric parts, it simplifies the feature recognition of prismatic parts considerably.

In finding the *ghost volumes*, the first task is to find the stock dimensions. The set difference of the stock and part gives the *cavity volume*, as a sum total of discrete *ghost volumes*. Different strategies are followed to find out the *ghost volumes* for 2.5-D parts and 3-D parts.

The 2.5-D implementation identifies the *ghost volumes*, divides them into machinable features and checks the machinability of features with the data base of available tools. The 3-D implementation is carried out to identify the *ghost volumes* along with the tool accessibility analysis.

The *ghost volumes* for a 3-D part are identified by the division of the part into 2.5-D slices. End milling can be used to machine the *ghost volumes* from the stock as 2.5-D slices. This also facilitates the fixturing of stock. The *ghost volumes* are divided into two sets, one set constituting the *ghost volumes* above the central X-Y plane of stock and another set of *ghost volumes* below it. First set of *ghost volumes* can be machined in one setup of fixturing and second set in another setup. It is advisable to machine the set in which the total volume to be removed i.e. the union of *ghost volumes* is lesser first.

The present implementation has some limitations. The cutting tool database used in the machinability analysis of 2.5-D parts should be enhanced to contain a larger number and variety of tools and subsequently the machinability checking functions will get modified. The best approach would be to build an expert system to select machine tools, cutting tools and fixtures etc..

The present 3-D implementation considers cylindrical hollow primitives and non-cylindrical solid primitives. This needs to be extended to include all kinds of hollow primitives and solid primitives. The output of the present system is a set of *ghost*

volumes. Though end milling can be used to remove ghost volumes from the stock to manufacture the part, these ghost volumes require processing to extract machinable features. An expert system based on the lines described in Chapter-I can be built in the selection of machining operations, machine tools, cutting tools, jigs and fixtures.

The present implementation is heavily dependant on AutoCAD for its input. A 3-D part is assumed to be an ensemble of primitives, since AutoCAD allows to model a 3-D part in such a way. This model of a 3-D part does not contain the connectivity information about the primitives and any boolean operations done on the primitives.

Rendering of graphics is also incomplete in the present implementation of 3-D parts. Only the global ghost volumes are displayed using Turbo graphics. The rendering can be extended to display the local ghost volumes, holes and machinable features using different views.

Chang, T. C. and Joshi, S. The Automated Factory Hand Book, eds Cleland, D. and Bidanda, B., Tab books 1990 pp 367-401.

Recquicha, A.A.G., and Vandenbrande, J., Automated Systems for Process Planning and Part Programming, Artificial Intelligence and CIM ed Kusiak, A..

Houtzeel, A., Computer Assisted Process Planning, A First Step Towards Integration in Proceedings of the 18th Numerical Control Society Annual Meeting and Technical Conference, Dallas, Texas 1981.

Recquicha, A.A.G., Representation for Rigid Solids : Theory, Methods and Systems, Computer Surveys, Vol.12, No.4, December 1980, pp 437-464.

Opitz, H., A Classification system to describe Work pieces, pergamon press, Elmford, New York 1978.

Allen, K., Generative Process Planning System Using DCLASS Information System, Monograph 4, Computer Aided Manufacturing Laboratory, Brigham Young University, Utah, 1979.

OIR, Multiplan, Organization for Industrial Research, Inc., Waltham, Mass., 1983.

Wysk, R.A., An Automated Process Planning and Selection Program : APPAS, Ph.D. Thesis, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, 1977.

Tulkoff, J., Lockheed's GENPLAN in Proceedings of 18th Numerical Control Society Annual Meeting and Technical Conference, pp 417-421, Dallas, Texas, 1981.

Evershiem, H., and Esch, H., Automated Generation of Process Plans for Prismatic Parts, Annals of CIRP, Vol 32/1/83, pp 361-364, 1983.

Jakubowski, R., Syntactic Characterization of Machine Part Shapes, Cybernetics and Systems : An International Journal, Vol. 13, pp 1-24, 1982.

Descotte, Y. and Latombe, J.C., GARI : A Problem Solver that Plans to Machine Mechanical Parts, in Proceedings of International Joint Conference on Artificial Intelligence, pp 766-772, 1981.

13. Wang, H. P. and Wysk, R.A., *Expert system models for Process Planning, Artificial Intelligence : Manufacturing Theory and Practice* eds Kumara, S. T., Kashyap R. L., and Soyster A.L., Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, Georgia, 1988, pp 535-564.
14. Khoshnevis, B. and Chen, Q., *Integration of Process Planning and Scheduling Functions*, Journal of Intelligent Manufacturing, September 1990.
15. Woo, T. C., *Computer Aided Recognition of Volumetric Designs*, Advances in Computer Aided Manufacturing, pp 121-135, North-Holland Publishing Co., 1977.
16. CAM-I, 1981, *CAM_I, Illustrated Glossary of Work Piece form features*, Report No. R-80-ppp-02-1, Computer Aided Manufacturing international Inc., Arlington, Texas.
17. Dong, X, and Wozney, M. J. *Feature volume criterion for Computer Aided Process Planning, Geometric Modeling for Product Engineering*, Elsevier Science Publishres, North-Holland, IFIP, 1990.
18. Perng, D., Chen, Z., and Li, R., *Automatic 3D Machining Feature Extraction from 3D CSG Solid Input*, CAD, Vol. 22, No. 5, June 1990.
19. Woo, T. C., *Feature Extraction by Volume Decomposition*, Proceedings of Conference on CAD/CAM Technology in Mechanical Engineering 1982, pp 76-94
20. Lee, Y. U. and Fu, K. S., *Machine Understanding of CSG : Extraction and Unification of Manufacturing Features*, IEEE Computer Graphics and Applications, January 1987, pp 20-32.
21. Henderson, M. R. and Anderson, D.C., *Computer Recognition and Recognition of Form Features : A CAD/CAM Link*, Computers in Industry, 5, 1984, pp 329-339.
22. Kyprianou, L. K., *Shape Classification in Computer Aided Design*, Ph.D. Thesis, Christ College, University of Cambridge, Cambridge, U.K., 1983.
23. Joshi, S. and Chang, T. C., *Graph Based Heuristics for Feature Recognition of Machined Features from a 3D Solid model*, CAD Vol. 20, No. 2, March 1988.
24. Joshi, S., *Feature recognition and Geometric Reasoning for Some Process Planning Activities*, Geometric Modeling in Product Engineering, Elsevier Science Publishers, North-Holland, IFIP, 1990.

25. Ansaldi., and Falcidieno, B., *Extraction and Completing Feature Information in Process Planning Applications*, Geometric Modeling for Product Engineering, Elsevier Science Publishers, North-Holland, IFIP, 1990.
26. Chang, T. C. and Wysk, R. A., *An introduction to Automated Process Planning Systems*, Prentice Hall, Englewood, New Jersey, 1985.
27. Rogers, D.F., *Procedural Elements for Computer Graphics*, McGraw-Hill Book Company, New York, 1985, pp 179-185.
28. Ghose, A. and Mallik, A.K., *Manufacturing Science*, Affiliated East-West Press Private Limited, New Delhi, 1985.
29. *Production Technology*, hmt, Tata McGraw hill Publishing Company Ltd., New Delhi, 1980.

NOTES ON VIVA-VOCE

A question was raised about the title of the thesis. It was asked that the title does not reveal about the environment to which the *feature recognition* is addressed. The term *feature recognition*, in the present-day literature, implies the extraction of manufacturing features. Since the mostly used manufacturing process is *metal cutting* and the title was already a lengthy one, no changes were made in title.

Second question was about the extraction of ghost volumes in a 3-d prismatic component having intersecting features. It was explained that such a kind of part can be handled by the proposed methodology.

Third question was about the extension of the methodology for prismatic parts containing the instances of hemisphere, a new primitive. Since the hemisphere is a non-cuboid, it can be replaced by a minimum enclosing cuboid before scanning the 3-D part to divide it into 2.5-D slices. The approach can similarly be extended to other new primitives.

Other question was asked about the repetition of finding the ghost volumes for similar 2.5-D slabs, which is being done in the present methodology. Finding whether a 2.5-D slab matches with other 2.5-D slab is not a trivial task and it may turn out to be tantamount, in effort, to repeating the ghost volume evaluation.